Matemática Discreta
para Computação e Informática

Matemática Discreta para Computação e Informática apresenta os principais conceitos e resultados de Matemática Discreta, usando uma linguagem simples e acessível a qualquer aluno de graduação, mas sem descuidar do desenvolvimento do raciocínio, nem dos aspectos teórico-formais. Sempre que possível, as construções apresentadas são instanciadas em casos aplicados a Computação e Informática, nas mais variadas matérias e disciplinas, como, por exemplo, *Sistemas Operacionais*, *Bancos de Dados*, *Compiladores*, *Estruturas de Dados*, *Técnicas Digitais*, *Algoritmos*, *Complexidade de Algoritmos*, *Teoria da Computação*, *Linguagens Formais e Autômatos*, *Modelos para Concorrência*, *Semântica Formal*, *Teoria das Categorias*, *Programação*, *Paradigmas de Linguagens de Programação*, etc. Este é um livro-texto para disciplinas dos cursos de graduação em Computação e Informática, de acordo com as Diretrizes Curriculares do MEC, bem como um livro de referência para diversos aspectos da computação em geral.

Trata-se de um trabalho baseado em experiências letivas desenvolvidas nos cursos de Bacharelado em Ciência da Computação e de Engenharia da Computação na UFRGS. Embora o conteúdo básico da matéria Matemática Discreta seja relativamente estável (comparativamente com e evolução tecnológica de Computação e Informática), a abordagem dá ênfase às questões e aos problemas da atualidade, bem como às novas abordagens. O livro é autocontido e possui uma apresentação que facilita a adequação do texto aos objetivos propostos e à carga horária da disciplina. É ilustrado com cerca de 140 figuras, 220 exemplos detalhados, 250 exercícios em níveis crescentes de raciocínio e um detalhado índice remissivo com cerca de 720 entradas. O único pré-requisito é o conteúdo de Matemática visto no Ensino Médio, e a carga horária total recomendada varia de 60 a 90 horas/aula.

Editora Sagra Luzzatto

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

ISBN 85-241-0691-3
9 788524 106910
INFORMATICA

511.1
M543m
2.ed.
e.5

Paulo Blauth Menezes

Matemática Discreta para Computação e Informática

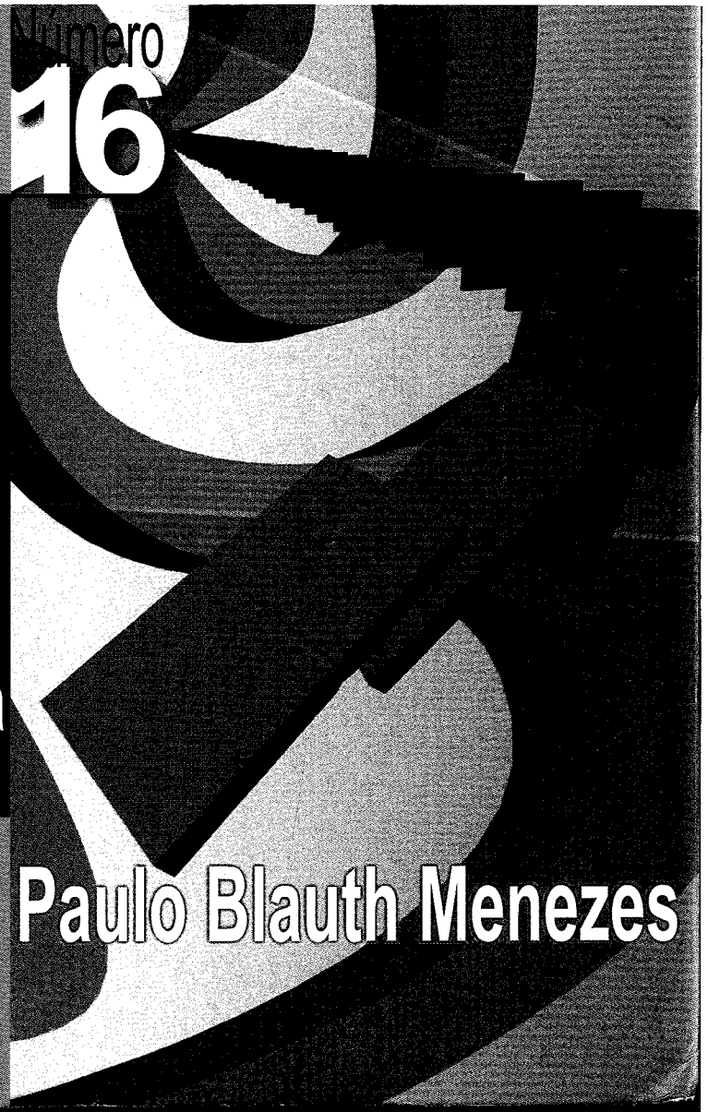
Série Livros Didáticos Número 16

2ª edição

Instituto de Informática da UFRGS

Editora Sagra Luzzatto

Matemática Discreta para Computação e Informática



Paulo Blauth Menezes

Série Livros Didáticos

Instituto de Informática
Universidade Federal do Rio Grande do Sul – UFRGS

Informática UFRGS

Diretor
Prof. Philippe Olivier Alexandre Navaux

Vice-Diretor
Prof. Otacílio José Carollo de Souza

Comissão Editorial
Prof. Paulo Blauth Menezes
Profª Ana Maria de Alencar Price
Profª Ingrid Jansch Pôrto
Prof. Luís C. Lamb
Prof. Ricardo Augusto da Luz Reis

Coordenador de Divulgação
Prof. Tiarajú Asmuz Diverio

Endereço
Universidade Federal do Rio Grande do Sul
Instituto de Informática
Av. Bento Gonçalves, 9500 - Bloco IV - Bairro Agronomia
Porto Alegre - RS - Brasil
CEP 91501-970 Caixa Postal 15064

Telefone: +55 (51) 3316-6165 e 3316-6168
Fax: +55 (51) 3316-7308

E-mail: livrosdidaticos@inf.ufrgs.br
http://www.inf.ufrgs.br (link "Publicações")

Paulo Blauth Menezes

Doutor em Matemática pelo IST/Universidade Técnica de Lisboa, Portugal. Mestre em Ciência da Computação e Licenciado em Matemática pela UFRGS. Professor e Pesquisador do Departamento de Informática Teórica do Instituto de Informática da UFRGS. Pesquisador CNPq.

Matemática Discreta
para Computação e Informática

Série Livros Didáticos
Instituto de Informática da UFRGS
Número 16 • 2ª edição

DEDALUS - Acervo - EACH

23000005510

Editora Sagra Luzzatto

ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES
BIBLIOTECA
5213
U.S.P.

511.1
M543m
2.ed.
e.5

S.m. 1492771

© de Paulo Blauth Menezes
1ª edição, 2004 | 2ª edição, 2005

Todos os direitos desta edição reservados à:

EDITORA SAGRA LUZZATTO S/A
Av. Independência, 1125 / 401
90035-077 Porto Alegre, RS
Telefone (51) 3227-5222 Fax 3227-4438
www.sagra-luzzatto.com.br
atendimento@sagra-luzzatto.com.br

Editor: Antônio Wenzel Luzzatto
Supervisão Editorial: Paulo Blauth Menezes
Capa: Malu Menezes (http://www.malumenezes.com)
Editoração: Instituto de Informática

Dados Internacionais de Catalogação na Publicação (CIP)
(Biblioteca do Instituto de Informática da UFRGS, Porto Alegre, RS)

Menezes, Paulo Blauth
Matemática Discreta para Computação e Informática / Paulo Blauth Menezes – Porto Alegre: Sagra Luzzatto / Instituto de Informática da UFRGS, 2005.
258 p.: il. - (Série Livros Didáticos, n.16)

1. Matemática Discreta : Ciência : Informática. 2. Informática Teórica. 3. Matemática : Computação I. Título. II. Série

É proibida a reprodução total ou mesmo parcial desta obra sem o consentimento prévio do autor e da editora.

Prefácio da Série

A série *Livros Didáticos*, do Instituto de Informática da Universidade Federal do Rio Grande do Sul, é inspirada na idéia de desenvolver material didático para disciplinas ministradas em cursos de graduação em Computação e Informática, ou seja, para os cursos de Bacharelado em Ciência da Computação, de Bacharelado em Sistemas de Informação, de Engenharia da Computação e de Licenciatura em Informática. A série é desenvolvida tendo em vista as Diretrizes Curriculares do MEC e é resultante da experiência dos professores do Instituto de Informática e dos colaboradores externos no ensino e na pesquisa.

Os primeiros títulos da série, iniciada em 1997, foram publicados no âmbito Projeto ArInPar – Aritmética Intervalar Paralela, financiado pelo ProTeM – CC CNPq/Fase II: *Fundamentos da Matemática Intervalar* e *Programando em Pascal XSC* (esgotado). Essas primeiras experiências serviram de base para os volumes subseqüentes, os quais se caracterizam como livros-texto para disciplinas básicas dos cursos de Computação e Informática. Em seus títulos mais recentes, a Série Livros Didáticos tem contado com a colaboração de professores externos que, em parceria com professores do referido Instituto, estão desenvolvendo livros de alta qualidade e de reconhecido valor didático.

O sucesso da experiência com esses livros, bem como a responsabilidade que cabe ao Instituto de Informática na formação de professores e pesquisadores em Computação e Informática, conduziu à ampliação da abrangência e à institucionalização da série.

Os seguintes títulos estão sendo correntemente publicados e têm sido adotados por diversas Universidades de vários Estados brasileiros:

- *Volume 1 – Fundamentos da Matemática Intervalar*, dos Professores Paulo Werlang de Oliveira, Tiarajú Asmuz Diverio e Dalcidio Moraes Claudio;
- *Volume 3 – Linguagens Formais e Autômatos*, do Professor Paulo Blauth Menezes (quinta edição);
- *Volume 4 – Projeto de Banco de Dados*, do Professor Carlos Alberto Heuser (quinta edição);
- *Volume 5 – Teoria da Computação: Máquinas Universais e Computabilidade*, dos Professores Tiarajú Asmuz Diverio e Paulo Blauth Menezes (segunda edição);
- *Volume 6 – Arquiteturas de Computadores Pessoais*, do Professor Raul Fernando Weber (segunda edição);
- *Volume 7 – Concepção de Circuitos Integrados*, organizado pelo Professor Ricardo Augusto da Luz Reis (segunda edição);
- *Volume 8 – Fundamentos de Arquitetura de Computadores*, do Professor Raul Fernando Weber (terceira edição);
- *Volume 9 – Implementação de Linguagens de Programação: Compiladores*, dos Professores Ana Maria de Alencar Price e Simão Sirineo Toscani (segunda edição);

- *Volume 10 – Tabelas: Organização e Pesquisa*, dos Professores Clesio Saraiva dos Santos e Paulo Alberto de Azeredo;
- *Volume 11 – Sistemas Operacionais*, dos Professores Rômulo Silva de Oliveira, Alexandre Silva Carissimi e Simão Sirineo Toscani (terceira edição);
- *Volume 12 – Teoria das Categorias para Ciência da Computação*, dos Professores Paulo Blauth Menezes e Edward Hermann Haeusler;
- *Volume 13 – Complexidade de Algoritmos*, dos Professores Laira Vieira Toscani e Paulo A. S. Veloso (segunda edição);
- *Volume 14 – Sistemas Operacionais e Programação Concorrente*, dos Professores Simão Sirineo Toscani, Rômulo Silva de Oliveira e Alexandre Silva Carissimi;
- *Volume 15 – Arquiteturas Paralelas*, dos Professores César A. F. De Rose e Philippe O. A. Navaux;
- *Volume 16 – Matemática Discreta para Computação e Informática*, do Professor Paulo Blauth Menezes (segunda edição);
- *Volume 17 – Fundamentos de Circuitos Digitais*, dos Professores Flávio Rech Wagner, Renato Perez Ribas e André Inácio Reis.

Em breve, um novo título será lançado:

- *Redes de Computadores*, dos Professores Alexandre Carissimi, Juergen Rochol e Lisandro Zambenedetti Granville.

Outros títulos, também compreendendo conteúdos de disciplinas de Bacharelado em Ciência da Computação, Bacharelado em Engenharia da Computação, Bacharelado em Sistemas de Informação e Licenciatura em Informática, encontram-se em preparação. Alguns estão sendo disponibilizados na forma de apostilas, objetivando verificação e validação dos textos.

Todos os livros têm em comum a preocupação em manter nível compatível com a elevada qualidade do ensino e da pesquisa desenvolvidos no âmbito do Instituto de Informática da UFRGS.

Mais detalhes sobre a Série Livros Didáticos, ou sobre os títulos publicados ou em preparação, podem ser encontrados na página do Instituto de Informática, http://www.inf.ufrgs.br, no link /Publicações/Série Livros Didáticos.

Prof. Paulo Blauth Menezes
Comissão Editorial da Série Livros Didáticos
Instituto de Informática da UFRGS
Abril de 2005

Este livro é dedicado a:

Maria Fernanda,

Maria Lúcia e

Maria Luiza

Prefácio do Autor

"Matemática Discreta para Computação e Informática" objetiva apresentar os principais conceitos e resultados de Matemática Discreta, usando uma linguagem simples e acessível a qualquer aluno de graduação, mas sem descuidar do desenvolvimento do raciocínio, nem dos aspectos matemático-formais. Sempre que possível, as construções apresentadas são instanciadas em casos aplicados a Computação e Informática, nas mais variadas matérias e disciplinas, como, por exemplo, *Sistemas Operacionais*, *Bancos de Dados*, *Compiladores*, *Estruturas de Dados*, *Técnicas Digitais*, *Algoritmos*, *Complexidade de Algoritmos*, *Teoria da Computação*, *Linguagens Formais e Autômatos*, *Modelos para Concorrência*, *Semântica Formal*, *Teoria das Categorias*, *Programação*, *Paradigmas de Linguagens de Programação*, etc. Este é um livro-texto para disciplinas dos cursos de graduação em Computação e Informática, de acordo com as Diretrizes Curriculares do MEC, bem como um livro de referência para diversos aspectos da computação em geral.

Trata-se de um trabalho baseado em experiências letivas desenvolvidas nos cursos de Bacharelado em Ciência da Computação e de Engenharia da Computação na Universidade Federal do Rio Grande do Sul. O livro é auto-contido, possui um texto simples, exemplos detalhados e exercícios em níveis crescentes de raciocínio.

A abordagem dá ênfase às questões e aos problemas da atualidade, bem como às novas abordagens dos Fundamentos da Computação, como as inspiradas em Teoria das Categorias. O texto é auto-contido e desenvolvido de forma natural e tem como único pré-requisito o conteúdo de Matemática visto no Ensino Médio.

Assim, o leitor que acompanhar satisfatoriamente o conteúdo tratado ao longo deste livro, além de:

a) Desenvolver a sua capacidade de raciocínio abstrato (lógico-matemático) como um todo;
b) Obter uma visão abrangente de uma parte significativa da Computação e Informática;

deverá ser capaz de:

c) Aplicar os conceitos básicos da Matemática Discreta como uma ferramenta Matemática para investigações e aplicações precisas em Computação e Informática;
d) Via Matemática Discreta, abordar problemas aplicados e enfrentar ou propor com naturalidade novas tecnologias.

Embora esta publicação seja abrangente e relativamete profunda (considerando o contexto a que se destina) não é objetivo cobrir todos os tópicos de Matemática Discreta. Assim, alguns temas como Análise Combinatória e Probabilidade Discreta não são desenvolvidos. Adicionalmente, apenas alguns tópicos de Teoria dos Grafos são introduzidos. A carga horária recomendada é de 60 a 90 horas, dependendo da formação dos alunos, domínio da matemática do Ensino Médio e do detalhamento dos exemplos e aplicações computacionais. Em particular, alguns temas relacionados com Computação e Informática são desenvolvidos em seções sugeridas como leitura complementar. Essas seções não constituem pré-requisito para outras seções, e seu estudo é, portanto, uma opção, facilitando a adequação do texto aos objetivos propostos e à carga horária da disciplina. De qualquer maneira, recomenda-se fortemente o desenvolvimento dos exercícios de

implementação sugeridos. Experiências demonstram que o entendimento dos alunos é significativamente reforçado e ampliado nesse caso.

Cálculo é barbada...

...eu faço Matemática Discreta !

Um dos motivacionais deste trabalho é expresso nessa imagem
estampada nas camisetas de alunos de Matemática Discreta da UFRGS (anos '90)

Entre os motivacionais do desenvolvimento deste trabalho, destacam-se:

a) *Disciplina especialmente difícil.* A Matemática Discreta é considerada especialmente difícil para o aluno, sendo muitas vezes classificada como mais difícil do que Cálculo (uma das maiores barreiras para alunos de diversos cursos). De fato, não é rara entre os alunos a idéia de que não é possível passar na disciplina logo na primeira vez em que é cursada. Assim, no desenvolvimento deste livro, houve uma grande preocupação com os aspectos didáticos, resultando em um texto simples e acessível, mas sem descuidar do desenvolvimento do raciocínio, nem dos aspectos matemático-formais;

b) *Expectativa tecnológica.* Freqüentemente, o aluno de Computação e Informática possui uma forte expectativa de estudos tecnológicos já no início do Curso. De fato, até o seu ingresso no Curso, poucos alunos têm uma noção clara da carga de disciplinas com ênfase teórico-formal. Assim, quando se deparam com um conjunto considerável de disciplinas com essa ênfase, tendem a considerar os estudos matemáticos como algo secundário ou de menor importância. Mesmo que o professor enfatize a importância da Matemática para o Curso e para a formação profissional, a idéia como um todo fica em um contexto muito abstrato. Uma conseqüência é que, mesmo que o aluno absorva o conteúdo desenvolvido, tende a esquecê-lo com rapidez, muitas vezes antes de aplicá-lo nas disciplinas subseqüentes. Assim, no desenvolvimento deste livro, o conteúdo matemático desenvolvido é constantemente instanciado em diversas matérias da Computação e Informática, não só para visualizar e entender como é aplicado, mas também para ajudar a fixar o conteúdo desenvolvido;

c) *Familiaridade com o conteúdo.* Muitos tópicos de Matemática Discreta são próximos do conteúdo desenvolvido no Ensino Médio, o qual, provavelmente, foi recém revisado pelo aluno quando do ingresso no Ensino Superior. Assim, a primeira impressão é de que se trata de uma revisão, com algum aprofundamento, de uma coletânea de tópicos já conhecidos, induzindo o aluno a um estudo relativamente superficial. O fato é que, na Matemática Discreta, a abrangência e a profundidade com que os assuntos são tratados são bem maiores, a abordagem é diferente (ênfase nos aspectos teórico-formais e no desenvolvimento do raciocínio), e muitos conceitos conhecidos são redefinidos para o contexto da Computação e Informática. Este livro destaca e diferencia o conteúdo (provavelmente) conhecido do novo, ressaltando, de forma direta ou indireta, quando não se trata de uma simples revisão aprofundada de tópicos já conhecidos;

d) *Ênfase dos estudos no conteúdo.* O conteúdo de Matemática Discreta é relativamente extenso e é desenvolvido com abrangência e profundidade. Tal fato tende a levar o aluno a centrar seu estudo no conteúdo, dando pouca atenção aos níveis mais elevados de raciocínio. A conseqüência é que, no meio do semestre letivo (ou até antes), muitos alunos se sentem subitamente perdidos, não acompanhando mais o desenvolvimento da disciplina. A questão fundamental é o entendimento de que, tão importante quanto o conteúdo, é o desenvolvimento da capacidade de raciocínio abstrato (lógico-matemático), o qual é fortemente explorado junto com o conteúdo. Ou seja, de certa forma, o conteúdo é usado como um meio para o desenvolvimento de um raciocínio abstrato. É importante observar que o desenvolvimento do raciocínio é obtido gradualmente, ao longo do tempo, como conseqüência de estudos regulares e sistemáticos, preferencialmente após cada aula ou tópico estudado. Conseqüentemente, estudos intensivos pouco antes de uma prova são muito pouco produtivos e, em geral, resultam na sensação do tipo: "sabia o conteúdo, mas não consegui resolver as questões". Assim, este livro apresenta o texto e os exercícios em níveis crescentes de raciocínio. Em particular, nos exercícios sobre um determinado tópico, a primeira metade enfatiza a fixação do conteúdo, ao passo que a segunda metade enfatiza o desenvolvimento do raciocínio.

Com o objetivo de manter o custo do livro acessível (fator importante para muitos estudantes), optou-se, para esta segunda edição, fazer apenas pequenas revisões, mantendo, sempre que possível, as mesmas numerações (páginas, seções, etc.) da primeira edição.

Material de apoio ou complementar que eventualmente venha a ser desenvolvido será disponibilizado no seguinte endereço da WEB:

http://teia.inf.ufrgs.br

Em particular, se o leitor for professor, sugere-se um contato direto para verificar eventual material de apoio específico.

Os autor agradece:

- aos alunos de pós-graduação Karina Girardi Roggia, Claudio Naoto Fuzitaki, Júlio Pereira Machado, Aline Malanovicz, Renata Gomes Wotter e à graduada Maria Lúcia Menezes;

- agradecimento especial à bolsista de iniciação científica Simone Bavaresco a qual contribuiu de forma direta para a viabilização deste trabalho;
- agradecimento muito especial ao aluno de pós-graduação Marnes Augusto Hoff, pelas excelentes contribuições, discussões, idéias (incluindo algumas originais) e cujo apoio foi fundamental ao longo de todo o trabalho;
- aos colegas Tiarajú Diverio, Laira Vieira Toscani, Daltro José Nunes e Luís Lamb pela amizade, apoio e incentivo recebidos para a viabilização deste e de outros trabalhos. A Laira, em particular, contribuiu com diversas revisões incorporadas nesta segunda edição;
- ao Amílcar Sernadas, ao Carlos Caleiro, ao Jaime Ramos, ao Pedro Resende e aos demais colegas do IST, pelo incentivo, apoio, inspiração, discussões, contribuições e amizade durante a minha estada em Portugal;
- à Liana Costi Nacul, Vilmar Trevisan, Maria Paula Fachin e ao Departamento de Matemática Pura a Aplicada do Instituto de Matemática da UFRGS pelo apoio e incentivo na realização deste trabalho;
- à Maria Medianeira, responsável por grande parte da experiência no ensino de Matemática Discreta nos Cursos de Computação e Informática do Instituto de Informática da UFRGS;
- a todos os alunos e ex-alunos das turmas de graduação do Instituto de Informática da UFRGS. Todos devem se considerar citados nominalmente neste espaço;
- aos irmãos João Pedro e João Paulo Leal pela amizade e apoio em alguns momentos difíceis. Em particular, ao João Pedro, pelas ações em benefício do desenvolvimento científico e tecnológico do Brasil, e do Instituto de Informática da UFRGS em particular, das quais este trabalho se beneficiou;
- aos amigos do Classic Car Club – RS (http://www.classiccarclub-rs.com.br), pelos momentos de descontração;
- ao CNPq, à CAPES e à FAPERGS, pelos auxílios financeiros aos projetos de pesquisa que viabilizaram a realização deste e de outros trabalhos.

Um agradecimento especial ao Instituto de Informática da UFRGS que tem apoiado esta série de *Livros Didáticos*.

Paulo Blauth Menezes
blauth@inf.ufrgs.br
http://www.inf.ufrgs.br/~blauth, http://teia.inf.ufrgs.br

Abril de 2005

Sumário

1 Introdução e Conceitos Básicos 1
1.1 Introdução à Matemática Discreta 1
1.2 Conceitos Básicos de Teoria dos Conjuntos 2
1.2.1 Conjuntos 2
1.2.2 Pertinência 4
1.2.3 Alguns Conjuntos Importantes 4
1.2.4 Conjuntos Finitos e Infinitos 5
1.2.5 Alfabetos, Palavras e Linguagens 5
1.2.6 Subconjunto e Igualdade de Conjuntos 6
1.2.7 Conjuntos nas Linguagens de Programação 8
1.3 Exercícios 10

2 Noções de Lógica e Técnicas de Demonstração 13
2.1 Lógica 14
2.1.1 Proposições 14
2.1.2 Conetivos 14
2.1.3 Fórmulas, Linguagem Lógica e Tabelas-Verdade 19
2.1.4 Lógica nas Linguagens de Programação 21
2.1.5 Tautologia e Contradição 23
2.1.6 Implicação e Equivalência 23
2.1.7 Quantificadores 25
2.2 Técnicas de Demonstração 29
2.2.1 Prova Direta 31
2.2.2 Prova por Contraposição 32
2.2.3 Prova por Redução ao Absurdo 32
2.3 Exercícios 33

3 Álgebra de Conjuntos 37
3.1 Diagramas de Venn 38
3.2 Paradoxo de Russell 40
3.3 Operações Não-Reversíveis 41
3.3.1 União 42
3.3.2 Intersecção 44
3.4 Operações Reversíveis 47
3.4.1 Complemento 47
3.4.2 Conjunto das Partes 50
3.4.3 Produto Cartesiano 52
3.4.4 União Disjunta 53
3.5 Relação entre Lógica e Álgebra de Conjuntos 55
3.6 Álgebra de Conjuntos nas Linguagens de Programação 57
3.7 Álgebra de Conjuntos e Teoria da Computação 59
3.8 Exercícios 63

4 Relações 67
4.1 Relação 68
4.2 Endorrelação como Grafo 71
4.3 Relação como Matriz 72
4.4 Relação Dual e Composição de Relações 73
4.4.1 Relação Dual 73
4.4.2 Composição de Relações 75
4.5 Tipos de Relações 77
4.5.1 Funcional e Injetora 78
4.5.2 Total e Sobrejetora 79
4.5.3 Monomorfismo e Epimorfismo 80
4.5.4 Isomorfismo 81
4.6 Banco de Dados Relacional 84
4.7 Rede de Petri 87
4.7.1 Modelo e Exemplos 87
4.7.2 Rede de Petri como Relação 88
4.8 Relações nas Linguagens de Programação 91
4.9 Exercícios 92

5 Funções Parciais e Totais 95
5.1 Função Parcial 96
5.1.1 Definição e Introdução 96
5.1.2 Função Parcial Dual 97
5.1.3 Composição de Funções Parciais 98
5.1.4 Restrição 99
5.2 Autômato Finito 100
5.2.1 Modelo e Exemplo 100
5.2.2 Autômato Finito como Função Parcial 102
5.2.3 Restrição de um Autômato Finito 103
5.2.4 Leitura Complementar 104
5.3 Função Total 104
5.3.1 Definição e Introdução 105
5.3.2 Exemplos Importantes de Funções 106
5.3.3 Função Dual 109
5.3.4 Composição de Funções 110
5.4 Construções Matemáticas como Funções 111
5.4.1 Relação como Função 111
5.4.2 Multiconjunto 111
5.4.3 Seqüência 112
5.4.4 Conjunto Indexado 113
5.5 Função de Hashing 114
5.6 Funções nas Linguagens de Programação 115
5.7 Linguagem de Programação Funcional 117
5.7.1 Haskell 117
5.7.2 Leitura Complementar 118
5.8 Exercícios 120

6 Endorrelações, Ordenação e Equivalência 125
6.1 Propriedades de uma Endorrelação 125
6.2 Fecho de uma Endorrelação 129
6.3 Ordenação 131
6.3.1 Relação de Ordem 132
6.3.2 Classificação de Dados 133
6.3.3 Diagrama de Hasse 135
6.3.4 Conjuntos Ordenados e Semântica de Sistemas Concorrentes 136
6.4 Equivalência e Partição 139
6.5 Exercícios 143

7 Cardinalidade de Conjuntos 147
7.1 Cardinalidade Finita e Infinita 147
7.2 Conjunto Contável e Não-Contável 148
7.3 Cardinalidade dos Conjuntos Não-Contáveis 150
7.4 Cardinal do Conjunto de Todos os Problemas Solucionáveis 152
7.5 Leitura Complementar: Máquina de Turing 152
7.5.1 Noção Intuitiva da Máquina de Turing 153
7.5.2 Modelo e Exemplo 153
7.5.3 Cardinal do Conjunto de Todas as Máquinas de Turing 157
7.6 Exercícios 158

8 Indução e Recursão 159
8.1 Princípio da Indução Matemática 159
8.2 Prova Indutiva 161
8.3 Segundo Princípio da Indução Matemática 163
8.4 Definição Indutiva 165
8.5 Expressões Regulares 167
8.6 Computações de um Autômato Finito 168
8.7 Leitura Complementar: Gramática e BNF 169
8.7.1 Gramática 170
8.7.2 BNF 172
8.8 Recursão 173
8.9 Leitura Complementar: Funções Recursivas Parciais 175
8.9.1 Substituição 175
8.9.2 Recursão Primitiva 176
8.9.3 Minimização 177
8.9.4 Função Recursiva Parcial 178
8.10 Exercícios 180

9 Álgebras e Homomorfismos 185
9.1 Operações Binárias 186
9.2 Propriedades das Operações Binárias 186
9.3 Grupóides, Semigrupos, Monóides, Grupos 188
9.4 Importantes Propriedades dos Monóides e Grupos 191
9.5 Homomorfismos 192
9.5.1 Homomorfismo de Grupóides e de Semigrupos 193
9.5.2 Homomorfismo de Monóides 197
9.5.3 Homomorfismo de Grupos 199
9.6 Monóide Livre Gerado e Fecho de Kleene 200
9.7 Grafos 202
9.8 Categorias 204
9.9 Exercícios 208

10 Reticulados e Álgebra Booleana 213
10.1 Limitantes de Conjuntos Parcialmente Ordenados 214
10.2 Reticulados 218
10.2.1 Reticulado como Relação de Ordem 218
10.2.2 Reticulado como Álgebra 221
10.3 Tipos Especiais de Reticulados 224
10.3.1 Reticulado Distributivo 224
10.3.2 Reticulado Limitado 225
10.3.3 Reticulado Complementado 226
10.4 Sub-Reticulado 227
10.5 Leitura Complementar: Primitivas para Programação Concorrente 229
10.6 Álgebra Booleana 231
10.7 Circuitos Lógicos 233
10.8 Homomorfismos 235
10.8.1 Homomorfismo de C.P.O. ou Função Monotônica 236
10.8.2 Homomorfismo de Reticulados 237
10.8.3 Homomorfismo de Álgebras Booleanas 238
10.9 Exercícios 239

11 Conclusões 245

12 Bibliografia 247

Índice Remissivo 251

1 Introdução e Conceitos Básicos

1.1 Introdução à Matemática Discreta

Praticamente qualquer estudo em Computação e Informática, teórico ou aplicado, exige como pré-requisito conhecimentos de diversos tópicos de Matemática. Tal fato é normalmente explicitado na maioria dos livros de Computação e Informática, sendo que alguns possuem um capítulo específico no qual tais tópicos são brevemente ou resumidamente introduzidos.

A importância da Matemática é explicitada nas Diretrizes Curriculares do MEC para Cursos de Computação e Informática [MEC 2005], como segue (a matéria Matemática é integrante da Área de Formação Básica):

> *A formação básica tem por objetivo introduzir as matérias necessárias ao desenvolvimento tecnológico da computação. O principal ingrediente desta área é a ciência da computação, que caracteriza o egresso como pertencente à área de computação. A maioria das matérias tecnológicas são aplicações da ciência da computação. São matérias de formação básica dos cursos da área de computação: a ciência da computação, a* ***matemática****, a física e eletricidade e a pedagogia.*

Especificamente em relação à matéria Matemática, o texto destaca que:

> *A matemática, para a área de computação, deve ser vista como uma ferramenta a ser usada na definição formal de conceitos computacionais (linguagens, autômatos, métodos etc). Os modelos formais permitem definir suas propriedades e dimensionar suas instâncias, dadas suas condições de contorno.*

Assim, este livro contém uma seleção de tópicos de Matemática os quais são essenciais para o estudo de Computação e Informática, tanto na Área de Formação Básica como na Área de Formação Tecnológica. Esta seleção de tópicos é comumente denominada de *Matemática Discreta*:

> *Considerando que a maioria dos conceitos computacionais pertencem ao domínio do discreto, a* ***matemática discreta*** *(ou também chamada álgebra abstrata) é fortemente empregada.*

Deve-se observar que não é objetivo desta publicação cobrir todos os tópicos de Matemática Discreta. Assim, alguns temas como *Análise Combinatória* e *Probabilidade Discreta* não são desenvolvidos. Adicionalmente, apenas alguns tópicos de *Teoria dos Grafos* são introduzidos.

Uma questão importante para o entendimento do que segue é a origem do termo Matemática Discreta. Intuitivamente falando, qualquer sistema computador possui limitações finitas em todos os seus principais aspectos como, por exemplo, tamanho da memória, número de instruções que pode executar, número de símbolos diferentes que pode tratar, etc. Assim, o estudo dos *conjuntos finitos* é fundamental.

O fato de um sistema computador possuir limitações finitas não implica necessariamente uma limitação ou pré-fixação de tamanhos máximos. Por exemplo, no que se refere à capacidade de armazenamento, um computador pode possuir unidades auxiliares como discos removíveis, fitas, etc. Portanto, para um correto entendimento de diversos aspectos computacionais, freqüentemente não é possível pré-fixar limites, o que implica tratar tais questões em um contexto infinito.

Entretanto, qualquer conjunto de recursos computacionais, finito ou infinito, é *contável* ou *discreto* (em oposição ao termo *contínuo*), no sentido em que seus elementos (recursos) podem ser *enumerados* ou seqüenciados (segundo algum critério) de tal forma que não existe um elemento entre quaisquer dois elementos consecutivos da enumeração. Por exemplo, o conjunto dos números naturais é obviamente contável. Um importante contra-exemplo é o conjunto dos números reais, o qual é *não-contável* ou *não-discreto*. Isso significa que existem conjuntos infinitos contáveis e conjuntos infinitos não-contáveis.

Assim, a *Matemática Discreta* possui como ênfase os estudos matemáticos baseados em conjuntos contáveis, finitos ou infinitos. Em oposição, a *Matemática do Continuum* possui como ênfase os estudos matemáticos baseados em conjuntos não-contáveis. Um importante exemplo de Matemática do *Continuum* para Computação e Informática é o *Cálculo Diferencial e Integral*.

Com o objetivo de manter o leitor de Computação e Informática motivado com o desenvolvimento do conteúdo, sempre que possível, para cada conceito de Matemática Discreta introduzido, são discutidas aplicações típicas de diversas áreas da Computação e Informática.

1.2 Conceitos Básicos de Teoria dos Conjuntos

O texto que segue introduz alguns conceitos básicos relativos à *Teoria dos Conjuntos* os quais, possivelmente, são do conhecimento do leitor. Neste caso, sugere-se uma rápida passagem para verificar as nomenclaturas e convenções adotadas ao longo do livro, bem como a leitura da exemplificação de seu uso em Computação e Informática.

1.2.1 Conjuntos

O conceito de conjunto é fundamental, pois praticamente todos os conceitos desenvolvidos em Computação e Informática, bem como os correspondentes resultados, são baseados em conjuntos ou construções sobre conjuntos.

Conjunto é uma estrutura que agrupa objetos e constitui uma base para construir estruturas mais complexas. Assim, informalmente, um conjunto é uma coleção, sem repetições e sem qualquer ordenação, de objetos denominados *elementos*. O termo "elemento" é usado de forma ampla e pode designar um objeto concreto ou abstrato. Neste contexto, um elemento é uma entidade básica a qual não é definida formalmente.

Definição 1.1 - Conjunto

Um *Conjunto* é uma coleção de zero ou mais objetos distintos, chamados *Elementos* do conjunto os quais não possuem qualquer ordem associada. ❑

EXEMPLO 1.1 - Conjuntos

a) As vogais a, e, i, o e u;
b) O par de sapatos preferido;
c) Os dígitos 0, 1, 2, 3, 4, 5, 6, 7, 8 e 9;
d) Todos os brasileiros;
e) Os números pares 0, 2, 4, 6,...
f) O personagem Snoopy, a letra a, a baía da Guanabara e o Pelé. ❑

Observe que um conjunto pode ser definido, listando-se todos os seus elementos (como "as vogais a, e, i, o e u") ou por propriedades declaradas (como "todos os brasileiros"). Adicionalmente, deve ficar claro que um conjunto não necessariamente é constituído por objetos que compartilham mesmas características ou propriedades (como em "o personagem Snoopy, a letra a, a baía da Guanabara e o Pelé").

A definição de um conjunto listando todos os seus elementos é denominada *denotação por extensão* e é dada pela lista de todos os seus elementos, em qualquer ordem (separados por vírgulas) e entre chaves como, por exemplo:

$$\text{Vogais} = \{a, e, i, o, u\}$$

Neste caso, Vogais denota o conjunto $\{a, e, i, o, u\}$.

A definição de um conjunto por propriedades é denominada *denotação por compreensão* como, por exemplo:

$$\text{Pares} = \{n \mid n \text{ é número par}\}$$

a qual é interpretada como:

o conjunto de todos os elementos n tal que n é número par

Assim, a forma geral de definição de um conjunto por propriedades é como segue:

$$\{x \mid p(x)\}$$

e é tal que um determinado elemento a é elemento deste conjunto se a propriedade p é verdadeira para a, ou seja, se p(a) é verdadeira. Por exemplo, para o conjunto:

$$B = \{x \mid x \text{ é brasileiro}\}$$

tem-se que Pelé é elemento de B e Bill Gates *não* é elemento de B.

Embora seja possível definir qualquer conjunto por compreensão, freqüentemente é conveniente especificar conjuntos de outra forma como, por exemplo:

$$\text{Dígitos} = \{0, 1, 2, 3, \ldots, 9\}$$
$$\text{Pares} = \{0, 2, 4, 6, \ldots\}$$

nos quais os elementos omitidos podem ser facilmente deduzidos do contexto.

EXEMPLO 1.2 - Conjuntos

a) $\text{Semana} = \{\text{seg, ter, qua, qui, sex, sab, dom}\}$
b) $\text{Duas Vogais} = \{\text{aa, ae, ai, ao, au, ea, ee, ei, eo, eu}, \ldots, \text{ua, ue, ui, uo, uu}\}$
c) $\{x \mid x = y^2 \text{ sendo que } y \text{ é número inteiro}\}$
o que corresponde ao conjunto $\{0, 1, 4, 9, 16, \ldots\}$ ❑

1.2.2 Pertinência

Se um determinado elemento a é elemento de um conjunto A, tal fato é denotado por:

$$a \in A$$

o qual é interpretado como segue:

a pertence *ao conjunto A*

Caso contrário, afirma-se que a *não* pertence ao conjunto A. Tal fato é denotado por:

$$a \notin A$$

EXEMPLO 1.3 - Pertence, Não-Pertence

a) Relativamente ao conjunto $\text{Vogais} = \{a, e, i, o, u\}$, tem-se que:

$a \in \text{Vogais}$
$h \notin \text{Vogais}$

b) Relativamente ao conjunto $B = \{x \mid x \text{ é brasileiro}\}$, tem-se que:

$\text{Pelé} \in B$
$\text{Bill Gates} \notin B$ ❑

1.2.3 Alguns Conjuntos Importantes

Um conjunto especialmente importante é o *conjunto vazio*, ou seja, o conjunto sem elementos { }, o qual é usualmente representado pelo seguinte símbolo:

$$\varnothing$$

EXEMPLO 1.4 - Conjunto Vazio

a) o conjunto de todos os brasileiros com mais de 300 anos;
b) o conjunto de todos os números os quais são simultaneamente pares e ímpares. ❑

Um tipo de conjunto quase tão importante como o vazio é o *conjunto unitário*, ou seja, um conjunto constituído por um único elemento. Portanto, existem infinitos conjuntos unitários. Entretanto, para muitas aplicações, pode-se usar qualquer conjunto unitário, ou seja, o fato importante é que o conjunto considerado possui um único elemento, sendo irrelevante qual é o elemento que o constitui. Nesse caso, um conjunto unitário fixado é usualmente denotado por 1.

EXEMPLO 1.5 - Conjunto Unitário

a) o conjunto constituído pelo jogador de futebol Pelé;
b) o conjunto de todos os números que são simultaneamente pares e primos;
c) $1 = \{*\}$ ❑

Os seguintes conjuntos (os quais devem ser do conhecimento do leitor) são importantes na Matemática em geral e na Computação e Informática em particular e possuem uma denotação universalmente aceita:

N *Conjunto dos Números Naturais*
Z *Conjunto dos Números Inteiros*
Q *Conjunto dos Números Racionais*
I *Conjunto dos Números Irracionais*
R *Conjunto dos Números Reais*

1.2.4 Conjuntos Finitos e Infinitos

Um conjunto pode possuir um número finito ou infinito de elementos. A definição formal de conjunto finito e infinito será apresentada adiante. Informalmente, um conjunto é dito:

a) *Conjunto finito* se pode ser denotado por extensão, ou seja, listando exaustivamente todos os seus elementos;
b) *Conjunto infinito*, caso contrário.

EXEMPLO 1.6 - Conjunto Finito, Conjunto Infinito

a) Os seguintes conjuntos são *finitos*:

$\varnothing$
$\{\varepsilon\}$
$\text{Vogais} = \{a, e, i, o, u\}$
$\text{Dígitos} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$\{\text{Snoopy, a, baía da Guanabara, Pelé}\}$
$A = \{x \in \mathbf{N} \mid x > 0 \text{ e } x < 4\}$
$B = \{x \mid x \text{ é brasileiro}\}$

b) Os seguintes conjuntos são *infinitos*:

$\mathbf{Z}$
$\mathbf{R}$
$\{x \in \mathbf{Z} \mid x \geq 0\}$
$\text{Pares} = \{y \mid y = 2x \text{ e } x \in \mathbf{N}\}$

Observe que o conjunto $A = \{x \in \mathbf{N} \mid x > 0 \ x < 4\}$ é finito, pois poderia ser representado por extensão, ou seja, $A = \{1, 2, 3\}$. Analogamente para o conjunto $B = \{x \mid x \text{ é brasileiro}\}$ (a representação por extensão é claramente possível, embora trabalhosa). Também, repare que o conjunto $\{x \in \mathbf{Z} \mid x \geq 0\}$ corresponde ao conjunto **N**. ❑

1.2.5 Alfabetos, Palavras e Linguagens

A noção de conjunto permite definir *linguagem*, um dos conceitos mais fundamentais em Computação e Informática. Para a definição de linguagem, é necessário antes introduzir os conceitos de *alfabeto* e de *cadeia de caracteres*. O estudo detalhado destes e de outros conceitos correlatos é realizado em algumas disciplinas como *Linguagens Formais* e *Compiladores*.

Definição 1.2 - Alfabeto

Um *Alfabeto* é um conjunto finito. Os elementos de um alfabeto são usualmente denominados de *símbolos* ou *caracteres*. ❑

Portanto, o conjunto vazio *é* um alfabeto, e um conjunto infinito *não* é um alfabeto.

Definição 1.3 - Palavra, Cadeia de Caracteres, Sentença

Uma *Palavra* ou *Cadeia de Caracteres* ou *Sentença* sobre um alfabeto é uma seqüência finita de símbolos (do alfabeto) justapostos. ❑

Portanto, uma cadeia sem símbolos é uma palavra válida, e o símbolo:

ε denota a *cadeia vazia*, *palavra vazia* ou *sentença vazia*.

Se Σ representa um alfabeto, então:

Σ^* denota o conjunto de todas as palavras possíveis sobre Σ

EXEMPLO 1.7 - Alfabeto, Palavra

a) Os conjuntos $\varnothing$ e $\{a, b, c\}$ são alfabetos;
b) O conjunto **N** *não* é um alfabeto;
c) ε é uma palavra sobre o alfabeto $\{a, b, c\}$;
d) ε é uma palavra sobre o alfabeto $\varnothing$;
e) a, e, i, o, u, ai, oi, ui e aeiou são exemplos de palavras sobre Vogais;
f) 1 e 001 são exemplos de palavras *distintas* sobre Dígitos;
g) $\{a, b\}^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$
h) $\varnothing^* = \{\varepsilon\}$

Definição 1.4 - Linguagem Formal

Uma *Linguagem Formal*, ou simplesmente *Linguagem*, é um conjunto de palavras sobre um alfabeto. ❑

EXEMPLO 1.8 - Linguagem Formal

Suponha o alfabeto $\Sigma = \{a, b\}$. Então:

a) O conjunto vazio e o conjunto formado pela palavra vazia são linguagens sobre Σ. Obviamente, $\varnothing \neq \{\varepsilon\}$;
b) O conjunto de *palíndromos* (palavras que têm a mesma leitura da esquerda para a direita e vice-versa) sobre Σ é um exemplo de linguagem (infinita):

$$\text{Palíndromos} = \{\varepsilon, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, \ldots\}$$ ❑

EXEMPLO 1.9 - Linguagens de Programação

As linguagens de programação como Pascal, C e Java são linguagens sobre o alfabeto constituído por letras, dígitos e alguns símbolos especiais (como espaço, parênteses, pontuação, etc.). Nesse caso, cada programa na linguagem corresponde a uma palavra sobre o alfabeto. Ou seja, uma linguagem de programação é definida por todos os seus programas possíveis. Portanto, Pascal, Java, C, bem como qualquer linguagem de programação de propósitos gerais, são conjuntos infinitos. ❑

Observação 1.10 - Compilador × Pertinência à Linguagem

Um *compilador* de uma linguagem de programação é um *software* que traduz um programa escrito na linguagem de programação (*linguagem fonte*) para um código executável no sistema computador (*linguagem objeto*). Em geral, um compilador é estruturado em duas grandes partes: análise (*análise léxica*, *análise sintática* e *análise semântica*) e síntese (*geração* e *otimização* de código executável). Resumidamente, a análise verifica se um dado programa fonte p é, de fato, um programa válido para a linguagem L em questão, ou seja, verifica se:

$$p \in L$$ ❑

1.2.6 Subconjunto e Igualdade de Conjuntos

Além da noção de pertinência já introduzida, outra noção fundamental da Teoria dos Conjuntos é a de *continência*, a qual permite introduzir os conceitos de *subconjunto* e de *igualdade de conjuntos*.

Se todos os elementos de um conjunto A também são elementos de um conjunto B, então se afirma que A está *contido* em B e denota-se por:

$$A \subseteq B$$

ou, alternativamente, que B *contém* A, e denota-se por:

$$B \supseteq A$$

Nesse caso ($A \subseteq B$ ou $B \supseteq A$), afirma-se que A é *subconjunto* de B.

Adicionalmente, se $A \subseteq B$, mas existe $b \in B$ tal que $b \notin A$, então se afirma que A está *contido propriamente* em B, ou que A é *subconjunto próprio* de B, e denota-se por:

$$A \subset B$$

Ou, alternativamente, que B *contém propriamente* A e denota-se por:

$$B \supset A$$

Quando não é fato que $A \subseteq B$ (respectivamente, $A \subset B$), é usual denotar como segue:

$$A \not\subseteq B \text{ (respectivamente, } A \not\subset B)$$

Outra terminologia usual, mas que não será adotada neste livro, é a seguinte:

- *contido amplamente* significando contido;
- *contido estritamente* significando contido propriamente.

EXEMPLO 1.11 - Continência, Subconjunto

a) $\{a, b\} \subseteq \{b, a\}$
b) $\{a, b\} \subseteq \{a, b, c\}, \{a, b\} \subset \{a, b, c\}$
c) $\{1, 2, 3\} \subseteq \mathbf{N}, \{1, 2, 3\} \subset \mathbf{N}$
d) $\mathbf{N} \subseteq \mathbf{Z}, \mathbf{N} \subset \mathbf{Z}$
e) $\varnothing \subseteq \{a, b, c\}, \varnothing \subset \{a, b, c\}$
f) $\varnothing \subseteq \mathbf{N}, \varnothing \subset \mathbf{N}$ ❑

Um conjunto especial e importante é o *conjunto universo*, normalmente denotado por **U**, o qual contém todos os conjuntos que estão sendo considerados. Ou seja, o conjunto universo define o "contexto de discussão" (e portanto, **U** *não* é um conjunto fixo). Uma vez definido o conjunto universo **U**, para qualquer conjunto A, tem-se que:

$$A \subseteq \mathbf{U}$$

Os conjuntos A e B são ditos *conjuntos iguais*, o que é denotado por:

$$A = B$$

se e somente se possuem exatamente os mesmos elementos. Formalmente, afirma-se que:

$$A = B \quad \text{se e somente se} \quad A \subseteq B \text{ e } B \subseteq A$$

EXEMPLO 1.12 - Igualdade de Conjuntos

a) $\{1, 2, 3\} = \{x \in \mathbf{N} \mid x > 0 \text{ e } x < 4\}$
b) $\mathbf{N} = \{x \in \mathbf{Z} \mid x \geq 0\}$
c) $\{1, 2, 3\} = \{3, 3, 3, 2, 2, 1\}$

Este último item ilustra claramente a definição de igualdade. De fato, é fácil verificar que:

$$\{1, 2, 3\} \subseteq \{3, 3, 3, 2, 2, 1\} \quad \text{e} \quad \{3, 3, 3, 2, 2, 1\} \subseteq \{1, 2, 3\}.$$

Observe que este exemplo também ilustra por que as repetições de elementos podem ser desconsideradas (se o leitor tiver alguma dúvida, revise a definição de continência). ❑

É importante distinguir claramente entre pertinência e continência. Sugere-se ler atentamente o exemplo que segue.

EXEMPLO 1.13 - Pertinência × Continência

Lembre-se de que os elementos de um conjunto podem ser quaisquer objetos. Em particular, podem ser conjuntos. Considere o conjunto $A = \{1, 2, 3, \varnothing, \{a\}, \{b, c\}\}$. Então (justifique):

a) $\{1\} \notin A, \{1\} \subseteq A$
b) $\varnothing \in A, \varnothing \subseteq A$
c) $\{a\} \in A, \{b, c\} \in A$
d) $\{1, 2, 3\} \notin A, \{1, 2, 3\} \subseteq A$ ❑

Observação 1.14 - Linguagem Formal × Conjunto de Todas as Palavras

Considere a Definição 1.4 - Linguagem Formal. Uma *linguagem formal* L sobre um alfabeto Σ pode alternativamente ser definida como um subconjunto de Σ*, ou seja:

$$L \subseteq \Sigma^* \qquad ❑$$

1.2.7 Conjuntos nas Linguagens de Programação

Como já introduzido, ao longo de todo o livro são exemplificadas aplicações de Matemática Discreta em Computação e Informática. Uma aplicação constantemente explorada é a relação entre os conceitos matemáticos desenvolvidos e suas implementações em linguagens de programação. Entretanto, como conhecimentos de linguagem de programação não são pré-requisitos deste livro, a exemplificação será ilustrativa e, portanto, não é detalhada, nem formal.

Com o objetivo de manter um entendimento crescente e coerente da aplicação da Matemática Discreta em linguagens de programação, o texto é centrado na linguagem Pascal, por diversas razões, com destaque para as seguintes:

- é uma linguagem desenvolvida objetivando o ensino de programação;
- é formalmente bem definida, o que facilita o seu estudo matemático;
- inspirou diversas linguagens de programação comerciais;
- é disponível em diversos tipos de sistemas computadores;
- é freqüentemente adotada como primeira linguagem de programação em cursos de Computação e Informática.

Para exemplificar conjuntos em linguagens de programação, é necessário antes introduzir o conceito de tipo de dado. Informalmente e resumidamente, um *tipo de dados* em Computação e Informática é um conjunto de objetos (dados) e certas operações sobre esses objetos. Considerando as limitações usuais dos sistemas computadores e objetivando manter a portabilidade dos *softwares*, algumas linguagens especificam os limites dos valores do tipo de dados, como os valores devem ser armazenados e como as operações devem ser processadas.

A grande maioria das linguagens de programação possui alguns tipos de dados predefinidos como, por exemplo:

- *Real* ou *Ponto Flutuante*
- *Inteiro*
- *Caractere*
- *Booleano* ou *Lógico*

Considerando as limitações físicas de representação de conjuntos infinitos ou de precisão de valores em um sistema computador (esse assunto normalmente é detalhado em disciplinas como *Arquitetura de Computadores* e *Matemática Computacional*), os tipos Real e Inteiro implementam um subconjunto próprio de **R** e de **Z**, respectivamente (bem como algumas operações tradicionais como adição, multiplicação, etc.).

O tipo Caractere implementa os caracteres usuais como letras e dígitos, bem como alguns símbolos especiais (parênteses, pontuação, aspas, etc.). O tipo Lógico implementa os valores lógicos `verdadeiro` e `falso`. Para ambos os tipos, são implementadas operações especiais, algumas das quais são discutidas ao longo deste livro.

Pode aparentar uma contradição, mas muitas linguagens de programação não possuem facilidades adequadas para definir e operar conjuntos. Em particular, Pascal oferece algum tratamento de conjuntos. De fato, em Pascal, pode-se definir tipos baseados em conjuntos *finitos*. Por exemplo, na última linha, o texto `'a'..'z'` denota o intervalo [a,...,z]):

```
cores set of (amarelo, vermelho, azul, branco, preto)
dias_semana set of (seg, ter, qua, qui, sex, sab, dom)
alfabeto set of 'a'..'z'
```

Adicionalmente, pode-se definir constantes e variáveis de um tipo conjunto. A definição de constantes de um tipo conjunto é realizada sempre por extensão, como, por exemplo:

```
[vermelho, amarelo, azul]
[ ]
[seg..dom]
[seg..sex]
['a', 'e', 'i', 'o', 'u']
```

os quais correspondem aos seguintes conjuntos, respectivamente:

$\{\text{vermelho, amarelo, azul}\}$
$\varnothing$
$\{\text{seg, ter, qua, qui, sex, sab, dom}\}$
$\{\text{seg, ter, qua, qui, sex}\}$
$\{a, e, i, o, u\}$

A definição de variáveis de um tipo conjunto é realizada listando quais nomes (das variáveis) correspondem a quais tipos. Por exemplo, no que segue, são declaradas (definidas) cinco variáveis de três tipos:

```
cores_primarias: cores
feriado, semana, trabalho: dias_semana
vogais: alfabeto
```

Assim, os seguintes trechos de programas em Pascal:

```
cores_primarias := [vermelho, amarelo, azul]
feriado := [ ]
semana := [seg..dom]
trabalho := [seg..sex]
vogais := ['a', 'e', 'i', 'o', 'u']
```

correspondem, na Teoria dos Conjuntos, aos seguintes conjuntos e suas correspondentes denotações:

$\text{cores_primarias} = \{\text{vermelho, amarelo, azul}\}$
$\text{feriado} = \varnothing$
$\text{semana} = \{\text{seg, ter, qua, qui, sex, sab, dom}\}$
$\text{trabalho} = \{\text{seg, ter, qua, qui, sex}\}$
$\text{vogais} = \{a, e, i, o, u\}$

Observe que, no trecho de programa acima, foi usado o símbolo "**:=**" e não "=" para associar a variável ao seu valor. De fato, em Pascal, bem como na maioria das linguagens de programação, o símbolo "=" é usado exclusivamente para verificar uma igualdade, e não para definir ou atribuir valores. Essa distinção objetiva facilitar a construção do correspondente *compilador*. No desenvolvimento de compiladores, o tratamento de questões sintáticas é bem mais simples do que o tratamento de questões semânticas. Por essa razão, o teste da igualdade e a atribuição são *sintaticamente* distinguidos.

As noções de igualdade, subconjunto (contido) e pertinência também podem ser especificadas em Pascal, usando-se a seguinte simbologia:

<= (continência)
= (igualdade)

Por exemplo, suponha os trechos de programas acima e considere os seguintes trechos (observe que, no que segue, o símbolo "=" é usado com o sentido de igualdade):

```
cores_primarias = [vermelho, amarelo, azul]
feriado = trabalho
trabalho <= semana
[sab, dom] <= trabalho
'a' in vogais
dom in trabalho
```

os quais correspondem às seguintes proposições sobre a Teoria dos Conjuntos:

$\text{cores_primarias} = \{\text{vermelho, amarelo, azul}\}$	(verdadeiro)
$\text{feriado} = \text{trabalho}$	(falso)
$\text{trabalho} \subseteq \text{semana}$	(verdadeiro)
$\{\text{sab, dom}\} \subseteq \text{trabalho}$	(falso)
$a \in \text{vogais}$	(verdadeiro)
$\text{dom} \in \text{trabalho}$	(falso)

1.3 Exercícios

Exercício 1.1 Para cada conjunto abaixo:

- descreva de forma alternativa (usando outra forma de notação);
- diga se é finito ou infinito.

a) Todos os números inteiros maiores que 10
b) $\{1, 3, 5, 7, 9, 11, \ldots\}$
c) Todos os países do mundo
d) A linguagem de programação Pascal

Exercício 1.2 Para $A = \{1\}$, $B = \{1, 2\}$ e $C = \{\{1\}, 1\}$, marque as afirmações corretas:

a) $A \subset B$ [ ]
b) $A \subseteq B$ [ ]
c) $A \in B$ [ ]
d) $A = B$ [ ]
e) $A \subset C$ [ ]

f) $A \subseteq C$ [ ]
g) $A \in C$ [ ]
h) $A = C$ [ ]
i) $1 \in A$ [ ]
j) $1 \in C$ [ ]
k) $\{1\} \in A$ [ ]
l) $\{1\} \in C$ [ ]
m) $\varnothing \notin C$ [ ]
n) $\varnothing \subseteq C$ [ ]

Exercício 1.3 Sejam $a = \{x \mid 2x = 6\}$ e $b = 3$. Justifique ou refute a seguinte afirmação:

$$a = b$$

Exercício 1.4 Quais são todos os subconjuntos dos seguintes conjuntos?

a) $A = \{a, b, c\}$
b) $B = \{a, \{b, c\}, D\}$ dado que $D = \{1, 2\}$

Exercício 1.5 O conjunto vazio está contido em qualquer conjunto (inclusive nele próprio)? Justifique a sua resposta.

Exercício 1.6 Todo conjunto possui um subconjunto próprio? Justifique a sua resposta.

Exercício 1.7 Sejam $A = \{0, 1, 2, 3, 4, 5\}$, $B = \{3, 4, 5, 6, 7, 8\}$, $C = \{1, 3, 7, 8\}$, $D = \{3, 4\}$, $E = \{1, 3\}$, $F = \{1\}$ e X um conjunto desconhecido. Para cada item abaixo, determine quais dos conjuntos A, B, C, D, E ou F podem ser iguais a X:

a) $X \subseteq A$ e $X \subseteq B$
b) $X \not\subseteq B$ e $X \subseteq C$
c) $X \not\subseteq A$ e $X \not\subseteq C$
d) $X \subseteq B$ e $X \not\subseteq C$

Exercício 1.8 Sejam A um subconjunto de B e B um subconjunto de C. Suponha que $a \in A$, $b \in B$, $c \in C$, $d \notin A$, $e \notin B$, $f \notin C$. Quais das seguintes afirmações são verdadeiras?

a) $a \in C$
b) $b \in A$
c) $c \notin A$
d) $d \in B$
e) $e \notin A$
f) $f \notin A$

Exercício 1.9 Marque os conjuntos que são alfabetos:

a) Conjunto dos números naturais [ ]
b) Conjunto dos números primos [ ]
c) Conjunto das letras do alfabeto brasileiro [ ]
d) Conjunto dos algarismos arábicos [ ]
e) Conjunto dos algarismos romanos [ ]
f) Conjunto $\{a, b, c, d\}$ [ ]
g) Conjunto das vogais [ ]
h) Conjunto das letras gregas [ ]

Exercício 1.10 Sejam $\Sigma = \{a, b, c, \ldots, z\}$ e $\text{Dígitos} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ alfabetos. Então:

a) Para cada um dos alfabetos abaixo, descreva o correspondente conjunto de todas as palavras:
   a.1) Σ
   a.2) Dígitos

b) Discuta as seguintes afirmações:
   b.1) Português é uma linguagem sobre Σ, ou seja, é um subconjunto de Σ^*
   Dica: Quais os símbolos usados para compor um texto em português?
   b.2) **N** é uma linguagem sobre Dígitos, ou seja, é um subconjunto de Dígitos*
   b.3) **N** = Dígitos*
   Dica: Como fica o caso da palavra vazia?

Exercício 1.11 Em que condições o conjunto de todos os palíndromos sobre um alfabeto constitui uma linguagem *finita*?

Exercício 1.12 Para que o leitor se convença plenamente da importância da Matemática Discreta para a Computação e Informática, sugere-se, como exercício complementar, duas pesquisas na *internet*, a saber:

a) Uma sobre Currículos de Cursos de Computação e Informática no mundo, e sua relação com Matemática Discreta. Observe que algumas vezes Matemática Discreta é denominada de Álgebra;
b) Outra sobre a importância da Matemática Discreta para a Computação e Informática e o detalhamento do porquê do termo "discreta".

Exercício 1.13 Mesmo que o leitor não tenha conhecimentos de linguagens de programação, é possível, com um rápido estudo, desenvolver programas simples em Pascal. Assim, sugere-se como exercício complementar de pesquisa, o desenvolvimento de um programa em Pascal que implemente os trechos de programas exemplificados.

2 Noções de Lógica e Técnicas de Demonstração

Lógica Matemática é básica para qualquer estudo em Computação e Informática e, em particular, para o estudo de Matemática Discreta. De fato, para desenvolver qualquer algoritmo e, conseqüentemente, qualquer *software* computacional, são necessários conhecimentos básicos de Lógica. Inclusive, existem linguagens de programação baseadas em Lógica, ou seja, desenvolvidas segundo o paradigma lógico, como, por exemplo, a linguagem Prolog.

As Diretrizes Curriculares do MEC para Cursos de Computação e Informática [MEC 2005] destacam que:

> *Lógica Matemática é uma ferramenta fundamental na definição de conceitos computacionais*

E, para algumas matérias da Área de Formação Tecnológica, como Inteligência Artificial, esse texto destaca, enfaticamente que:

> *Como base ao estudo da Inteligência Artificial são* ***imprescindíveis*** *conhecimentos de Lógica Matemática, ...*

Lógica permite definir a noção de *teorema*. Em um primeiro momento, pode parecer estranho que teoremas e suas correspondentes demonstrações sejam fundamentais para a Computação e Informática. Entretanto, em computação, um teorema freqüentemente pode ser visto como um problema a ser implementado computacionalmente, e a sua correspondente *demonstração*, pode ser vista como uma solução computacional, ou seja, um *algoritmo*. Adicionalmente, o algoritmo que soluciona o problema, prova-se, sempre funciona!

Para ilustrar a importância desse assunto para a Computação e Informática, David Parnas, importante pesquisador internacional e um dos pioneiros da *Engenharia de Software*, afirmou em uma palestra no XIII SBES – Simpósio Brasileiro de Engenharia de *Software* que:

> *o maior avanço da Engenharia de* Software *nos últimos dez anos foram os provadores de teoremas*

O principal objetivo deste capítulo é introduzir, resumidamente, os principais conceitos e a terminologia de *lógica matemática* e de *técnicas de demonstração de teoremas* necessários para o estudo de Matemática Discreta, portanto, esta não é uma abordagem ampla, nem detalhada. Tal abordagem foge do escopo deste livro e normalmente é desenvolvida em uma disciplina específica de Lógica. Relacionado à lógica, é introduzida a *Hipótese de Church* a qual é uma *premissa básica* para toda a Computação e Informática.

2.1 Lógica

O texto que segue é centrado em Lógica Booleana. Entende-se por *Lógica Booleana* ou *Lógica de Boole* o estudo dos princípios e métodos usados para distinguir sentenças verdadeiras de falsas. George Boole (inglês, 1815-1864) foi um dos precursores do estudo da Lógica.

2.1.1 Proposições

Definição 2.1 - Proposição
Uma *Proposição* é uma construção (sentença, frase, pensamento) à qual se pode atribuir juízo. No caso da *Lógica Matemática*, o tipo de juízo é o verdadeiro-falso, ou seja, o interesse é na "verdade" das proposições. ❑

Em Lógica Matemática, a forma tradicional de tratar com a "verdade" é considerar dois *valores-verdade* V e F (*verdadeiro* e *falso*, respectivamente) e estipular que as proposições só podem assumir esses dois valores. Para uma dada proposição p, denota-se por:

$$V(p)$$

o valor-verdade de p.

EXEMPLO 2.1 - Proposição

a) São exemplos de proposições:

Brasil é um país
Buenos Aires é a capital do Brasil
$3+4>5$
$7-1=5$

Para cada uma dessas proposições, o valor-verdade é como segue:

V(Brasil é um país) = V
V(Buenos Aires é a capital do Brasil) = F
$V(3+4>5) = V$
$V(7-1=5) = F$

b) *Não* são exemplos de proposições:

Vá tomar banho.
Que horas são?
Parabéns! ❑

2.1.2 Conetivos

As proposições introduzidas até o momento são ditas *proposições atômicas* ou simplesmente *átomos*, no sentido em que não podem ser decompostas em proposições mais simples.

Entretanto, é possível construir proposições mais complexas compondo proposições, usando *operadores lógicos*, também denominados de *conetivos (lógicos)*.

EXEMPLO 2.2 - Proposições Compostas
Nas seguintes proposições (compostas), os conetivos lógicos estão representados em negrito:

a) *Windows* é um sistema operacional, **e** Pascal é uma linguagem de programação;
b) Vou comprar um PC **ou** um MAC;
c) Linux **não** é um *software* livre;
d) **Se** choverem canivetes, **então** todos os alunos estarão aprovados em Matemática Discreta;
e) $A = B$ **se e somente se** ($A \subseteq B$ **e** $B \subseteq A$). ❑

Claramente, proposições compostas podem ser usadas para construir novas proposições compostas, como no caso do último item no EXEMPLO 2.2 - Proposições Compostas. O mesmo exemplo ilustra os cinco conetivos que serão estudados (e, ou, não, se-então e se-somente-se).

Negação

Já foi visto que uma dada proposição p ou é verdadeira ou é falsa. A *negação* de uma proposição é construída, introduzindo-se a palavra *não* de forma apropriada ou prefixando-se a proposição por "não é fato que" (ou expressão equivalente).

EXEMPLO 2.3 - Negação
Considere as seguintes proposições:

Brasil é um país;
Linux é um *software* livre;
$3+4>5$

A negação dessas proposições é como segue, respectivamente:

Brasil não é um país
Linux não é um *software* livre;
Não é fato que $3+4>5$ ❑

Se p denota uma proposição, então a negação de p é denotada por:

$$\neg p \quad \text{ou} \quad \sim p$$

a qual é lida como:

"não p"

sendo interpretada como segue:

- se p é verdadeira, então $\neg p$ é falsa;
- se p é falsa, então $\neg p$ é verdadeira.

Uma *tabela-verdade* é uma tabela que descreve os valores lógicos de uma proposição em termos das possíveis combinações dos valores lógicos das proposições componentes e dos conetivos usados. Assim, para cada combinação de valores-verdade e de conetivos, a tabela-verdade fornece o valor-verdade da expressão resultante.

Definição 2.2 - Negação
Dada uma proposição lógica p, a semântica da *Negação* $\neg p$ é dada pela tabela-verdade ilustrada na Figura 2.1. ❑

p	¬p
V	F
F	V

Figura 2.1 Tabela-verdade: negação

Conjunção

A conjunção de duas proposições p e q, denotada por:

$$p \wedge q$$

a qual é lida:

"p e q"

reflete uma noção de simultaneidade para ser verdadeira. Assim, a proposição composta $p \wedge q$ é:

- verdadeira, apenas quando p e q são simultaneamente verdadeiras;
- falsa, em qualquer outro caso.

Definição 2.3 - Conjunção
Dadas duas proposições lógicas p e q, a semântica da *Conjunção* $p \wedge q$ é dada pela tabela-verdade ilustrada na Figura 2.2. ❑

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

Figura 2.2 Tabela-verdade: conjunção

Relativamente à tabela-verdade da conjunção ilustrada na Figura 2.2, observe que, para expressar todas as combinações possíveis de valores lógicos das proposições p e q, foram necessárias quatro linhas (quantas linhas seriam necessárias para expressar a combinação de todos os valores lógicos possíveis de n proposições?).

EXEMPLO 2.4 - Conjunção
Sugere-se observar a tabela ilustrada na Figura 2.2, durante a leitura deste exemplo, procurando justificar por que a proposição composta, em cada um dos itens abaixo, é verdadeira ou falsa:

a) Verdadeira: *Windows* é um sistema operacional, **e** Pascal é uma linguagem de programação;
b) Falsa: *Windows* é um sistema operacional, **e** Pascal é uma planilha eletrônica;
c) Falsa: *Windows* é um editor de textos, **e** Pascal é uma linguagem de programação;
d) Falsa: *Windows* é um editor de textos, **e** Pascal é uma planilha eletrônica. ❑

Disjunção

A disjunção de duas proposições p e q, denotada por:

$$p \vee q$$

a qual é lida:

"p ou q"

reflete a noção de que pelo menos uma (eventualmente duas) das proposições componentes deve ocorrer para que a resultante seja verdadeira. Assim, a proposição composta $p \vee q$ é:

- verdadeira, quando pelo menos uma das proposições é verdadeira;
- falsa, somente quando simultaneamente p e q são falsas.

Definição 2.4 - Disjunção
Dadas duas proposições lógicas p e q, a semântica da *Disjunção* $p \vee q$ é dada pela tabela-verdade ilustrada na Figura 2.3. ❑

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

Figura 2.3 Tabela-verdade: disjunção

EXEMPLO 2.5 - Disjunção
Sugere-se observar a tabela ilustrada na Figura 2.3, durante a leitura deste exemplo, procurando justificar por que a proposição composta, em cada um dos itens abaixo, é verdadeira ou falsa:

a) Verdadeira: *Windows* é um sistema operacional **ou** Pascal é uma linguagem de programação;
b) Verdadeira: *Windows* é um sistema operacional **ou** Pascal é uma planilha eletrônica;
c) Verdadeira: *Windows* é um editor de textos **ou** Pascal é uma linguagem de programação;
d) Falsa: *Windows* é um editor de textos **ou** Pascal é uma planilha eletrônica. ❑

Condição

A condição envolvendo duas proposições p e q, denotada por:

$$p \rightarrow q$$

a qual é lida:

"se p então q"

reflete a noção de que, a partir de uma *premissa* verdadeira (ou seja, p é verdadeira), obrigatoriamente deve-se chegar a uma *conclusão* verdadeira (ou seja, q é verdadeira), para que a proposição composta $p \rightarrow q$ seja verdadeira. Entretanto, partindo de uma premissa falsa, qualquer conclusão pode ser considerada. Assim, a proposição composta $p \rightarrow q$ é:

- falsa, quando p é verdadeira e q é falsa;
- verdadeira, caso contrário.

Definição 2.5 - Condição
Dadas duas proposições lógicas p e q, a semântica da *Condição* $p \rightarrow q$ é dada pela tabela-verdade ilustrada Figura 2.4. ❑

p	q	$p \rightarrow q$
V	V	V
V	F	F
F	V	V
F	F	V

Figura 2.4 Tabela-verdade: condição

EXEMPLO 2.6 - Condição
Sugere-se observar a tabela ilustrada na Figura 2.4, durante a leitura deste exemplo, procurando justificar por que a proposição composta, em cada um dos itens abaixo, é verdadeira ou falsa:

a) Verdadeira: **se** *Windows* é um sistema operacional, **então** Pascal é uma linguagem de programação;
b) Falsa: **se** *Windows* é um sistema operacional, **então** Pascal é uma planilha eletrônica;
c) Verdadeira: **se** *Windows* é um editor de textos, **então** Pascal é uma linguagem de programação;
d) Verdadeira: **se** *Windows* é um editor de textos, **então** Pascal é uma planilha eletrônica. ❑

Bicondição

A bicondição envolvendo duas proposições p e q, denotada por:

$$p \leftrightarrow q$$

a qual é lida:

"p se e somente se q"

reflete a noção de condição "nos dois sentidos", ou seja, considera simultaneamente:

- sentido de "ida": p é premissa e q é conclusão;
- sentido de "volta": q é premissa e p é conclusão.

Portanto, considerando a noção de condição já introduzida e considerando que esta é "nos dois sentidos", a proposição composta $p \leftrightarrow q$ é:

- verdadeira, quando p e q são ambas verdadeiras ou ambas falsas;
- falsa, quando as proposições p e q possuem valor-verdade distintos.

Definição 2.6 - Bicondição
Dadas duas proposições lógicas p e q, a semântica da *Bicondição* $p \leftrightarrow q$ é dada pela tabela-verdade ilustrada na Figura 2.5. ❑

p	q	$p \leftrightarrow q$
V	V	V
V	F	F
F	V	F
F	F	V

Figura 2.5 Tabela-verdade: bicondição

EXEMPLO 2.7 - Bicondição
Sugere-se observar a tabela ilustrada na Figura 2.5, durante a leitura deste exemplo, procurando justificar por que a proposição composta, em cada um dos itens abaixo, é verdadeira ou falsa:

a) Verdadeira: *Windows* é um sistema operacional **se e somente se** Pascal é uma linguagem de programação;
b) Falsa: *Windows* é um sistema operacional **se e somente se** Pascal é uma planilha eletrônica;
c) Falsa: *Windows* é um editor de textos **se e somente se** Pascal é uma linguagem de programação;
d) Verdadeira: *Windows* é um editor de textos **se e somente se** Pascal é uma planilha eletrônica. ❑

2.1.3 Fórmulas, Linguagem Lógica e Tabelas-Verdade

Fórmulas Lógicas ou simplesmente *Fórmulas* são as palavras da *Linguagem Lógica*. O conceito de fórmula é formalmente introduzido adiante, quando do estudo da *Definição Indutiva*, mas pode ser facilmente entendido como sendo uma sentença lógica corretamente construída sobre o alfabeto cujos símbolos são conetivos ($\wedge$, $\vee$, $\rightarrow$, etc.), parênteses, identificadores (p, q, r, etc.), constantes, etc. Se uma fórmula contém variáveis, então esta *não* necessariamente possui valor-verdade associado. Ou seja, seu valor lógico depende do valor-verdade das sentenças que substituem as variáveis na fórmula.

EXEMPLO 2.8 - Fórmulas
Suponha que p, q e r são sentenças variáveis. Então, são fórmulas:

a) Os valores-verdade constantes V e F
b) Qualquer proposição
c) p, q e r
d) $\neg p$, $p \wedge q$, $p \vee q$, $p \rightarrow q$ e $p \leftrightarrow q$
e) $p \vee (\neg q)$
f) $(p \wedge \neg q) \rightarrow F$
g) $\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$
h) $p \vee (q \wedge r) \leftrightarrow (p \vee q) \wedge (p \vee r)$ ❑

Com o objetivo de possibilitar a redução do número de parênteses e, conseqüentemente, simplificar visualmente as fórmulas, a seguinte ordem de precedência entre os conetivos é convencionada. A ordem é a seguinte:

1. Conetivos entre parênteses, dos mais internos para os mais externos;
2. Negação ($\neg$)

3. Conjunção ($\wedge$) e disjunção ($\vee$)
4. Condição ($\rightarrow$)
5. Bicondição ($\leftrightarrow$)

EXEMPLO 2.9 - Precedência de Conetivos

a) $p \vee (\neg q)$ é equivalente a $p \vee \neg q$
b) $(p \wedge \neg q) \rightarrow F$ é equivalente a $p \wedge \neg q \rightarrow F$
c) $\neg(p \wedge q) \leftrightarrow (\neg p \vee \neg q)$ é equivalente a $\neg(p \wedge q) \leftrightarrow \neg p \vee \neg q$
d) para a fórmula $p \vee (q \wedge r) \leftrightarrow (p \vee q) \wedge (p \vee r)$ qualquer omissão de parênteses resulta em ambigüidade (por quê?). ❑

As *tabelas-verdade* foram introduzidas quando da definição dos conetivos. Entretanto, como construir uma tabela-verdade de uma dada fórmula? Lembre-se de que a tabela-verdade deve explicitar todas as combinações possíveis dos valores lógicos das fórmulas atômicas componentes. Observe que:

- cada fórmula atômica não-constante pode assumir dois valores lógicos: V e F. Obviamente, uma fórmula atômica *constante* possui um valor-verdade fixo (V ou F);
- portanto, na tabela-verdade da negação ilustrada na Figura 2.1 duas linhas são suficientes para expressar os valores lógicos possíveis;
- para as tabelas com duas fórmulas atômicas (não-constantes), como as da conjunção (Figura 2.2) e da condição (Figura 2.4), são necessárias quatro linhas, ou seja, 2^2 possíveis combinações dos valores lógicos;
- de fato, é fácil verificar que, para n fórmulas atômicas (não-constantes), são necessárias 2^n linhas na tabela-verdade para expressar todas as combinações possíveis de valores lógicos.

Tal fato, bem como a técnica de construção de tabelas-verdade, é ilustrado nos exemplos que seguem.

EXEMPLO 2.10 - Tabela-Verdade

Os passos de construção da tabela-verdade para a fórmula $p \vee \neg q$ são ilustrados na Figura 2.6, da esquerda para a direita. Observe que:

- a tabela possui $4 = 2^2$ linhas, pois se trata de duas fórmulas atômicas;
- as duas primeiras colunas expressam as combinações possíveis de p e q;
- a terceira coluna é introduzida e corresponde à negação de q, ou seja, à fórmula $\neg q$;
- a quarta coluna corresponde à disjunção de p com $\neg q$, ou seja, $p \vee \neg q$, a qual contém o resultado desejado. ❑

p	q
V	V
V	F
F	V
F	F

p	q	$\neg q$
V	V	F
V	F	V
F	V	F
F	F	V

p	q	$\neg q$	$p \vee \neg q$
V	V	F	V
V	F	V	V
F	V	F	F
F	F	V	V

Figura 2.6 Passos de construção de uma tabela-verdade

EXEMPLO 2.11 - Tabela-Verdade

A fórmula $p \wedge \neg q \rightarrow F$ possui 3 proposições atômicas. Entretanto, como a fórmula atômica F é constante, não é considerada no cálculo do número de linhas. Assim, a tabela-verdade possui $4 = 2^2$ e é ilustrada na Figura 2.7. Observe que não foi introduzida uma coluna para o valor constante F, pois a sua introdução seria redundante (a coluna conteria somente F).

p	q	$\neg q$	$p \wedge \neg q$	$p \wedge \neg q \rightarrow F$
V	V	F	F	V
V	F	V	V	F
F	V	F	F	V
F	F	V	F	V

Figura 2.7 Tabela-verdade

EXEMPLO 2.12 - Tabela-Verdade

A fórmula $p \vee (q \wedge r) \leftrightarrow (p \vee q) \wedge (p \vee r)$ possui 3 fórmulas atômicas não-constantes. Portanto, a tabela-verdade possui $8 = 2^3$ linhas e é ilustrada na Figura 2.8. Observe que a construção da tabela respeitou a ordem de precedência definida. ❑

p	q	r	$q \wedge r$	$p \vee (q \wedge r)$	$p \vee q$	$p \vee r$	$(p \vee q) \wedge (p \vee r)$	$p \vee (q \wedge r) \leftrightarrow (p \vee q) \wedge (p \vee r)$
V	V	V	V	V	V	V	V	V
V	V	F	F	V	V	V	V	V
V	F	V	F	V	V	V	V	V
V	F	F	F	V	V	V	V	V
F	V	V	V	V	V	V	V	V
F	V	F	F	F	V	F	F	V
F	F	V	F	F	F	V	F	V
F	F	F	F	F	F	F	F	V

Figura 2.8 Tabela-verdade

2.1.4 Lógica nas Linguagens de Programação

Já foi comentado que, em geral, as linguagens de programação possuem o tipo de dado lógico ou booleano predefinido.

No caso da linguagem Pascal, o tipo de dado é denominado `boolean`, e os correspondentes valores lógicos V e F são denotados por `true` e `false`, respectivamente. Assim, por exemplo, a declaração (definição) das variáveis p, q e r deste tipo é como segue:

```
p, q, r: boolean
```

Também já foi comentado que as noções de igualdade e continência (entre conjuntos) e de pertinência (de um elemento a um conjunto) são expressões que resultam em valores lógicos. Da mesma forma, as seguintes relações entre expressões aritméticas também resultam em valores lógicos:

= (igual)
< (menor)
<= (menor ou igual)
> (maior)
>= (maior ou igual)

Por exemplo, os seguintes trechos de programas em Pascal resultam em valores lógicos (qual o valor lógico resultante?):

```
7 - 1 = 5
n + 1 > n
```

A linguagem Pascal (assim como a maioria das linguagens de programação) possui os seguintes conetivos lógicos:

`not` (negação)
`and` (conjunção)
`or` (disjunção)
`<=` (condição)
`=` (bicondição)

EXEMPLO 2.13 - Programa em Pascal

Suponha que é desejado desenvolver um programa em Pascal capaz de calcular e informar o valor lógico da fórmula $p \vee (q \wedge r)$ para quaisquer valores de p, q e r fornecidos. Assim, o programa necessita ler os valores de p, q e r, calcular o valor lógico da fórmula para os valores lidos e imprimir o resultado.

Um programa para o problema é apresentado na Figura 2.9. Observe o seguinte:

- a primeira linha define o nome do programa (`valor_logico`) e informa que os procedimentos predefinidos `input` e `output` do sistema serão usados para entradas (leituras) e saídas (impressões), respectivamente;
- a segunda linha define as variáveis p, q e r as quais são do tipo `boolean`;
- entre as palavras `begin` e `end` são especificados os comandos (definem as ações);
- o comando de leitura (`read`) lê os valores lógicos de p, q e r a serem considerados;
- o comando `if-then-else` tem a seguinte semântica: se a expressão lógica após a palavra `if` for verdadeira, então o comando após a palavra `then` é executado; senão, o comando após a palavra `else` é executado;
- portanto, para os valores de p, q e r lidos, se o valor-verdade de $p \vee (q \wedge r)$ for verdadeiro, o texto *verdadeiro* é impresso; senão, o texto *falso* é impresso. ❑

```
program valor_logico (input, output);
var p, q, r: boolean;
begin
        read (p, q, r);
        if    p or (q and r)
        then  write('verdadeiro')
        else  write('falso')
end.
```

Figura 2.9 Programa em Pascal para calcular o valor lógico de $p \vee (q \wedge r)$

2.1.5 Tautologia e Contradição

Definição 2.7 - Tautologia, Contradição

Seja w uma fórmula. Então:

a) w é dita uma *Tautologia* se w é verdadeira, ou seja, se for verdadeira para todas as combinações possíveis de valores de sentenças variáveis;
b) w é dita uma *Contradição* se w é falsa, ou seja, se for falsa para todas as combinações possíveis de valores de sentenças variáveis. ❑

EXEMPLO 2.14 - Tautologia, Contradição

Suponha p uma fórmula. Considere a tabela-verdade ilustrada na Figura 2.10. Então:

a) A fórmula $p \vee \neg p$ é uma tautologia;
b) A fórmula $p \wedge \neg p$ é uma contradição. ❑

p	¬p	p ∨ ¬p	p ∧ ¬p
V	F	V	F
F	V	V	F

Figura 2.10 Tabela-verdade: tautologia e contradição

2.1.6 Implicação e Equivalência

Os conetivos de condição e bicondição induzem as relações de *implicação* e de *equivalência* entre fórmulas, respectivamente. A importância destas relações pode ser resumida como segue:

a) Relação de implicação: está diretamente relacionada com o conceito de *teorema*, como será visto adiante;
b) Relação de equivalência: permite definir a noção de "mesmo significado" entre duas fórmulas (sintaticamente) diferentes.

Definição 2.8 - Relação de Implicação

Sejam p e q duas fórmulas. Então p *implica* q, denotado por:

$$p \Rightarrow q \quad \text{ou} \quad p \models q$$

se e somente se:

$$p \rightarrow q \text{ é uma tautologia.}$$ ❑

EXEMPLO 2.15 - Relação de Implicação

Considere a tabela-verdade ilustrada na Figura 2.11. Então, valem as seguintes implicações (procure interpretar os nomes dados a cada uma das implicações):

a) *Adição*

$$p \Rightarrow p \vee q$$

b) *Simplificação*

$$p \wedge q \Rightarrow p$$ ❑

p	q	p ∨ q	p → (p ∨ q)	p ∧ q	(p ∧ q) → p
V	V	V	V	V	V
V	F	V	V	F	V
F	V	V	V	F	V
F	F	F	V	F	V

Figura 2.11 Tabela-verdade: adição e simplificação

Definição 2.9 - Relação de Equivalência
Sejam p e q duas fórmulas. Então p *é equivalente a* q, denotado por:

$$p \Leftrightarrow q$$

se e somente se:

$$p \leftrightarrow q \text{ é uma tautologia.}$$ ❑

EXEMPLO 2.16 - Relação de Equivalência
Considere a fórmula $p \vee (q \wedge r) \leftrightarrow (p \vee q) \wedge (p \vee r)$ apresentada anteriormente e a correspondente tabela-verdade ilustrada na Figura 2.8. Então, vale (observe a relação de equivalência):

$$p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$$

Este exemplo mostra que a distributividade do conetivo ou sobre o conetivo e é verdadeira *sempre* (sugere-se como exercício verificar se o conetivo e distribui-se sobre o conetivo ou). ❑

Os exemplos de equivalência que seguem são importantes para o estudo das *Técnicas de Demonstração* as quais são tratadas adiante. Assim, sugere-se uma especial atenção e um correto entendimento destes exemplos.

EXEMPLO 2.17 - Bicondição x Condição
Considere a tabela-verdade ilustrada na Figura 2.12. Então, vale:

$$p \leftrightarrow q \Leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$$

Este exemplo mostra formalmente por que a bicondição pode ser expressa por duas condições: "ida" e "volta". ❑

p	q	p ↔ q	p → q	q → p	(p → q) ∧ (q → p)	(p ↔ q) ↔ (p → q) ∧ (q → p)
V	V	V	V	V	V	V
V	F	F	F	V	F	V
F	V	F	V	F	F	V
F	F	V	V	V	V	V

Figura 2.12 Tabela-verdade: Bicondição x Condição

EXEMPLO 2.18 - Contraposição
Considere a tabela-verdade ilustrada na Figura 2.13. Então, vale a seguinte equivalência, conhecida por *contraposição*:

$$p \rightarrow q \Leftrightarrow \neg q \rightarrow \neg p$$ ❑

p	q	¬p	¬q	p → q	¬q → ¬p	p → q ↔ ¬q → ¬p
V	V	F	F	V	V	V
V	F	F	V	F	F	V
F	V	V	F	V	V	V
F	F	V	V	V	V	V

Figura 2.13 Tabela-verdade: contraposição

EXEMPLO 2.19 - Redução ao Absurdo
Considere a tabela-verdade ilustrada na Figura 2.14. Então, vale a seguinte equivalência, conhecida por *redução ao absurdo*:

$$p \rightarrow q \Leftrightarrow p \wedge \neg q \rightarrow F$$ ❑

p	q	¬q	p → q	p ∧ ¬q	p ∧ ¬q → F	p → q ↔ p ∧ ¬q → F
V	V	F	V	F	V	V
V	F	V	F	V	F	V
F	V	F	V	F	V	V
F	F	V	V	F	V	V

Figura 2.14 Tabela-verdade: redução ao absurdo

2.1.7 Quantificadores

Considere a seguinte sentença:

$$n > 1$$

Claramente, dependendo do valor de n considerado, esta sentença pode assumir o valor verdadeiro ou falso. Ou seja, para cada valor de n considerado, esta sentença é uma proposição diferente. Tal idéia induz a definição de proposição sobre um conjunto de valores. Adicionalmente, para uma dada proposição sobre um conjunto de valores, freqüentemente é desejável quantificar os valores a serem considerados.

Proposição Sobre um Conjunto

Definição 2.10 - Proposição sobre um Conjunto
Uma *Proposição Sobre* A é uma proposição cujo valor lógico depende do elemento $x \in A$ considerado. ❑

EXEMPLO 2.20 - Proposição sobre um Conjunto
São proposições sobre o conjunto **N**:

a) $n > 1$
b) $n! < 10$
c) $n + 1 > n$
d) 2n é ímpar

Quais proposições são verdadeiras para qualquer $n \in \mathbf{N}$? ❑

Uma proposição p a qual descreve alguma propriedade de um elemento $x \in A$ é usualmente denotada por:

$$p(x)$$

Toda proposição p sobre A determina dois conjuntos, como segue:

a) *Conjunto verdade* de p

$$\{x \in A \mid p(x) \text{ é verdadeira}\}$$

b) *Conjunto falsidade* de p

$$\{x \in A \mid p(x) \text{ é falsa}\}$$

EXEMPLO 2.21 - Conjuntos Verdade e Falsidade

Suponha o conjunto **N**. Então:

a) Para a proposição $n > 1$:

$\{2, 3, 4, \ldots\}$ é o conjunto verdade;
$\{0, 1\}$ é o conjunto falsidade;

b) Para a proposição $n! < 10$:

$\{0, 1, 2, 3\}$ é o conjunto verdade;
$\{n \in \mathbf{N} \mid n > 3\}$ é o conjunto falsidade;

c) Para a proposição $n + 1 > n$:

N é o conjunto verdade (é o próprio conjunto universo);
$\varnothing$ é o conjunto falsidade;

d) Para a proposição "2n é ímpar":

$\varnothing$ é o conjunto verdade;
N é o conjunto falsidade (é o próprio conjunto universo). ❑

Assim, uma proposição p sobre A é uma:

a) *Tautologia* se p(x) é verdadeira para qualquer $x \in A$, ou seja, o conjunto verdade é A;
b) *Contradição* se p(x) é falsa para qualquer $x \in A$, ou seja, o conjunto falsidade é A. ❑

EXEMPLO 2.22 - Tautologia, Contradição

Suponha o conjunto universo **N**. Então:

a) A fórmula $n! < 10$ não é tautologia nem contradição. De fato, por exemplo:

para $n = 0$, a fórmula é verdadeira
para $n = 4$, a fórmula é falsa

b) A fórmula $n + 1 > n$ é uma tautologia. De fato, o conjunto verdade é o conjunto universo **N**;
c) A fórmula "2n é ímpar" é uma contradição. De fato, o conjunto falsidade é o próprio conjunto universo **N**. ❑

Quantificador

Com freqüência, para uma determinada proposição p(x), é desejável quantificar os valores de x que devem ser considerados. Assim, os seguintes quantificadores são usados em Lógica (suponha uma proposição p(x)):

a) *Quantificador universal*, simbolizado por $\forall$, o qual, quando associado a uma proposição p(x), é denotado alternativamente como segue:

$$(\forall x \in A)(p(x)) \quad (\forall x \in A)\, p(x) \quad \forall x \in A,\ p(x)$$

b) *Quantificador existencial*, simbolizado por $\exists$, o qual, quando associado a uma proposição p(x), é denotado alternativamente como segue:

$$(\exists x \in A)(p(x)) \quad (\exists x \in A)\, p(x) \quad \exists x \in A,\ p(x)$$

Quando é claro sobre qual conjunto de valores a proposição está definida, uma expressão quantificada $(\forall x \in A)\, p(x)$ (respectivamente, $(\exists x \in A)\, p(x)$) pode ser simplesmente denotada como segue, respectivamente:

$$(\forall x)(p(x)) \quad (\forall x)\, p(x) \quad \forall x,\ p(x)$$
$$(\exists x)(p(x)) \quad (\exists x)\, p(x) \quad \exists x,\ p(x)$$

Uma leitura de uma proposição quantificada $(\forall x \in A)\, p(x)$ (respectivamente, $(\exists x \in A)\, p(x)$) é como segue, respectivamente:

"qualquer x, p(x)" ou "para todo x, p(x)"
"existe pelo menos um x tal que p(x)" ou "existe x tal que p(x)"

Como a leitura induz, o valor-verdade de uma proposição quantificada é como segue:

a) $(\forall x \in A)\, p(x)$ é verdadeira, se p(x) for verdadeira para *todos* os elementos de A;
b) $(\exists x \in A)\, p(x)$ é verdadeira, se p(x) for verdadeira para *pelo menos um* elemento de A.

Definição 2.11 - Quantificador Universal, Quantificador Existencial

Seja p(x) uma proposição lógica sobre um conjunto A. Então:

a) *Quantificador Universal*: a proposição $(\forall x \in A)\, p(x)$ é:

- verdadeira, se o conjunto verdade for igual ao conjunto A;
- falsa, caso contrário;

b) *Quantificador Existencial*: a proposição $(\exists x)\, p(x)$é:

- verdadeira, se o conjunto verdade for não-vazio;
- falsa, caso contrário. ❑

EXEMPLO 2.23 - Quantificador Universal, Quantificador Existencial

Procure justificar o valor-verdade de cada uma das seguintes proposições quantificadas:

a) $(\forall n \in \mathbf{N})(n < 1)$ é falsa
$(\exists n \in \mathbf{N})(n < 1)$ é verdadeira

b) $(\forall n \in \mathbf{N})(n! < 10)$ é falsa
$(\exists n \in \mathbf{N})(n! < 10)$ é verdadeira

c) $(\forall n \in \mathbf{N})(n + 1 > n)$ é verdadeira
$(\exists n \in \mathbf{N})(n + 1 > n)$ é verdadeira

d) $(\forall n \in \mathbf{N})$(2n é par) é verdadeira
$(\exists n \in \mathbf{N})$(2n é par) é verdadeira

Observe que, nos itens acima, sempre que uma proposição quantificada universalmente é verdadeira, a mesma proposição, mas quantificada existencialmente, também é verdadeira. Esta observação vale sempre? ❑

Claramente, a noção de uma proposição $p(x)$ sobre um conjunto pode ser generalizada, resultando em uma proposição p a qual descreve alguma propriedade de elementos $x_1 \in A_1$, $x_2 \in A_2, \ldots, x_n \in A_n$. Tal proposição é usualmente denotada por:

$$p(x_1, x_2, \ldots, x_n)$$

Portanto, cada um dos elementos $x_1, x_2, \ldots, x_n$ pode ser individualmente quantificado. Neste caso, a ordem dos quantificadores existencial e universal pode alterar o valor-verdade da proposição. Por exemplo, para o conjunto universo $\mathbf{N}$, vale:

$(\forall n)(\exists m)(n < m)$ é verdadeira
$(\exists m)(\forall n)(n < m)$ é falsa

De fato:

a) Para $(\forall n)(\exists m)(n < m)$, dado *qualquer* valor n, é possível encontrar (*existe*) pelo menos um m que satisfaz a desigualdade. Por exemplo, basta tomar $m = n + 1$. Assim, claramente, para qualquer número natural n, vale a proposição $n < n + 1$, que é trivialmente verdadeira;
b) A proposição $(\exists m)(\forall n)(n < m)$ afirma que *existe* um número natural maior que *qualquer* outro. Sabe-se que tal número natural não existe. Portanto, a proposição é falsa.

Observação 2.12 - Existe pelo menos um × Existe um único
É muito comum quantificar existencialmente de forma única, ou seja, de tal forma que existe um elemento e este é *único*. Portanto, não é uma quantificação existencial usual na qual pode existir mais de um. Tal quantificador é usualmente simbolizado por $\exists!$. Assim, por exemplo (compare com o EXEMPLO 2.23 - Quantificador Universal, Quantificador Existencial):

$(\exists! n \in \mathbf{N})(n < 1)$ é verdadeira
$(\exists! n \in \mathbf{N})(n! < 10)$ é falsa
$(\exists! n \in \mathbf{N})(n + 1 > n)$ é falsa
$(\exists! n \in \mathbf{N})(2n$ é par$)$ é falsa

Prova-se que o quantificador $\exists!$ pode equivalentemente ser representado usando os quantificadores universal e existencial e os conetivos usuais, como abaixo (o primeiro e o segundo termos da conjunção significam *existe* e *é único*, respectivamente). A equivalência não será provada:

$$(\exists! x)\, p(x) \Leftrightarrow (\exists x)\, p(x) \wedge (\forall x)(\forall y)(\,(p(x) \wedge p(y) \rightarrow x = y)\,) \qquad \square$$

Negação de Proposições Quantificadas

A negação de uma proposição quantificada é intuitiva. De fato, suponha a seguinte proposição quantificada:

$$(\forall x \in A)\, p(x)$$

cujo valor-verdade (já foi visto) é como segue:

$(\forall x \in A)\, p(x)$ é verdadeira, se $p(x)$ for verdadeira para *todos* os elementos de A

Assim, a negação significa que *não* é verdadeira para *todos* os elementos de A, ou seja, que *existe* pelo menos um x tal que não é fato que $p(x)$, e portanto:

$$(\exists x \in A)\, \neg p(x)$$

Logo:

$$\neg(\,(\forall x \in A)\, p(x)\,) \Leftrightarrow (\exists x \in A)\, \neg p(x)$$

Fazendo um raciocínio análogo, conclui-se que:

$$\neg(\,(\exists x \in A)\, p(x)\,) \Leftrightarrow (\forall x \in A)\, \neg p(x)$$

EXEMPLO 2.24 - Negação de Proposições Quantificadas
Considere o EXEMPLO 2.23 - Quantificador Universal, Quantificador Existencial. Procure justificar cada uma das seguintes equivalências:

a) $\neg(\,(\forall n \in \mathbf{N})(n < 1)\,) \Leftrightarrow (\exists n \in \mathbf{N})(n \geq 1) \Leftrightarrow V$
$\neg(\,(\exists n \in \mathbf{N})(n < 1)\,) \Leftrightarrow (\forall n \in \mathbf{N})(n \geq 1) \Leftrightarrow F$
b) $\neg(\,(\forall n \in \mathbf{N})(n! < 10)\,) \Leftrightarrow (\exists n \in \mathbf{N})(n! \geq 10) \Leftrightarrow V$
$\neg(\,(\exists n \in \mathbf{N})(n! < 10)\,) \Leftrightarrow (\forall n \in \mathbf{N})(n! \geq 10) \Leftrightarrow F$
c) $\neg(\,(\forall n \in \mathbf{N})(n + 1 > n)\,) \Leftrightarrow (\exists n \in \mathbf{N})(n + 1 \leq n) \Leftrightarrow F$
$\neg(\,(\exists n \in \mathbf{N})(n + 1 > n)\,) \Leftrightarrow (\forall n \in \mathbf{N})(n + 1 \leq n) \Leftrightarrow F$
d) $\neg(\,(\forall n \in \mathbf{N})(2n$ é par$)\,) \Leftrightarrow (\exists n \in \mathbf{N})(2n$ não é par$) \Leftrightarrow F$
$\neg(\,(\exists n \in \mathbf{N})(2n$ é par$)\,) \Leftrightarrow (\forall n \in \mathbf{N})(2n$ não é par$) \Leftrightarrow F$ $\square$

Claramente, a noção de negação pode ser estendida para proposições que dependem de n elementos, os quais podem ser individualmente quantificados, como ilustrado no exemplo a seguir.

EXEMPLO 2.25 - Negação de Proposições Quantificadas
Considere as seguintes proposições quantificadas:

$(\forall n)(\exists m)(n < m)$ é verdadeira
$(\exists m)(\forall n)(n < m)$ é falsa

A negação dessas proposições é, respectivamente:

$(\exists n)(\forall m)(n \geq m)$ é falsa
$(\forall m)(\exists n)(n \geq m)$ é verdadeira $\square$

2.2 Técnicas de Demonstração

Um *teorema* é uma proposição do tipo:

$$p \rightarrow q$$

a qual prova-se ser verdadeira sempre (tautologia), ou seja, que:

$$p \Rightarrow q$$

As proposições p e q são denominadas *hipótese* e *tese*, respectivamente. É usual denominar *corolário* um teorema que é uma conseqüência quase direta de um outro já demonstrado (ou seja, cuja prova é trivial ou imediata). Adicionalmente, um teorema auxiliar que possui um resultado importante para a prova de um outro é usualmente denominado *lema*.

Teoremas são fundamentais em Computação e Informática e, em particular, no estudo de Matemática Discreta. Por exemplo, um teorema permite verificar se uma determinada implementação é correta. Sob este ponto de vista, um teorema pode ser visto como um algoritmo que, prova-se, sempre funciona. De fato, muitos dos teoremas apresentados ao longo do livro refletem essa idéia.

Antes de iniciar uma demonstração, é fundamental identificar claramente a hipótese e a tese. Por exemplo, considere o seguinte teorema:

0 é o único elemento neutro da adição em $\mathbf{N}$

Uma reescrita, identificando claramente a hipótese e a tese, é como segue:

se 0 é elemento neutro da adição em $\mathbf{N}$,
então 0 é o único elemento neutro da adição em $\mathbf{N}$

É importante observar que, na demonstração de que, de fato, $p \Rightarrow q$, a hipótese p é *suposta* verdadeira. Conseqüentemente, a hipótese *não* deve ser demonstrada. De fato, todas as teorias possuem um conjunto de premissas (hipóteses) que são *supostas* verdadeiras e sobre as quais todo o raciocínio é construído. A Teoria dos Conjuntos, como introduzida neste livro, é baseada em uma premissa que é a noção de elemento, a qual é simplesmente suposta. Algumas abordagens também consideram a noção de conjunto como sendo uma premissa. A própria Computação e Informática é construída sobre uma importante premissa, conhecida como *Hipótese de Church*.

Observação 2.13 - Hipótese de Church × Computação e Informática
Um dos conceitos mais fundamentais da Computação e Informática é o de *algoritmo*, também denominado *procedimento efetivo* ou *função computável*. Intuitivamente, um algoritmo é:

uma seqüência finita de instruções, as quais podem ser realizadas mecanicamente, em um tempo finito

Claramente, tal intuição não pode corresponder a um conceito formal de algoritmo. De fato, no início do século XX, muitos pesquisadores dedicaram-se a formalizar tal conceito. Assim, diversas formalizações matemáticas foram desenvolvidas. Em particular, em 1936, Alan Turing propôs um modelo conhecido como *Máquina de Turing*, o qual é apresentado no Capítulo 7 - Cardinalidade de Conjuntos.

Ainda em 1936, Alonzo Church apresentou a *Hipótese de Church*, a qual afirma que:

qualquer função computável pode ser processada por uma Máquina de Turing, ou seja, existe um procedimento expresso na forma de uma Máquina de Turing capaz de processar a função

Contudo, como a noção intuitiva de algoritmo não é matematicamente precisa, é impossível demonstrar formalmente se a Máquina de Turing é, efetivamente, o mais genérico dispositivo de computação. Entretanto, foi mostrado que todos os demais modelos propostos possuem, no máximo, a mesma capacidade computacional dessa máquina, o que reforça a Hipótese de Church.

Desde então, para todo o desenvolvimento subseqüente, a Hipótese de Church é *suposta* e, conseqüentemente, é uma *premissa básica* para toda a Computação e Informática. Observe que, se algum dia for encontrado algum formalismo mais geral do que a Máquina de Turing, pela semântica do conetivo $\rightarrow$, os estudos desenvolvidos sobre a Hipótese de Church continuam válidos (por quê?). O estudo da Máquina de Turing, da Hipótese de Church e dos conceitos correlatos é desenvolvido pela *Teoria da Computação*. ❑

É usual que um teorema seja apresentado na forma $p \leftrightarrow q$. Entretanto, no EXEMPLO 2.17 - Bicondição × Condição, foi visto que:

$$p \leftrightarrow q \Leftrightarrow (p \rightarrow q) \wedge (q \rightarrow p)$$

Assim, uma técnica usual para demonstrar que $p \leftrightarrow q$ é provar em separado a "ida" $p \rightarrow q$ e a "volta" $q \rightarrow p$.

Para um determinado teorema $p \rightarrow q$, existem diversas técnicas para provar (demonstrar) que, de fato, $p \Rightarrow q$. Destacam-se as seguintes *técnicas de demonstração*:

a) *Prova Direta*;
b) *Prova por Contraposição*;
c) *Prova por Redução ao Absurdo* ou simplesmente *Prova por Absurdo*;
d) *Prova por Indução*.

A prova por indução é uma aplicação particular do *Princípio da Indução Matemática*, a qual será estudada em capítulo específico. Os demais tipos de prova são descritos a seguir. Destaque-se que, ao longo de todo o livro, cada demonstração é um exemplo de uso de tais técnicas. Seguindo este raciocínio, os exercícios de demonstração também estão distribuídos ao longo de todo o livro.

De qualquer forma, para qualquer técnica de demonstração, especial atenção deve ser dada aos quantificadores. Por exemplo, para provar a proposição:

$$(\forall x \in A)\ p(x)$$

é necessário provar $p(x)$ para todo $x \in A$. Portanto, mostrar um determinado elemento $a \in A$ não é uma prova, mas sim um exemplo, o que, neste caso, *não* constitui uma prova válida para todos os elementos de A. Já no seguinte caso:

$$(\exists x \in A)\ p(x)$$

basta provar que existe pelo menos um $a \in A$ tal que $p(a)$ é verdadeira. Ou seja, contrariamente ao caso acima (quantificador universal), neste caso um exemplo *é* uma prova.

2.2.1 Prova Direta

Definição 2.14 - Prova Direta
Uma prova é dita *Prova Direta* ou *Demonstração Direta* quando pressupõe verdadeira a hipótese e, a partir desta, prova ser verdadeira a tese. ❑

EXEMPLO 2.26 - Prova Direta
Considere o teorema:

a soma de dois números pares é um número par

ou seja, reescrevendo na forma de $p \rightarrow q$:

se n *e* m *são dois números pares quaisquer, então* $n+m$ *é um número par*

Inicialmente, lembre-se de que qualquer número par n pode ser definido como $n=2r$, para algum natural r.

Suponha que n e m são dois números pares. Então existem $r, s \in \mathbf{N}$ tais que:

$$n=2r \quad \text{e} \quad m=2s$$

Então:

$$n+m=2r+2s=2(r+s)$$

Como a soma de dois números naturais $r+s$ é um número natural, vale $n+m=2(r+s)$.

Logo, $n+m$ é um número par. ❑

2.2.2 Prova por Contraposição

A prova por contraposição baseia-se no seguinte resultado (denominado de *contraposição*), o qual foi verificado no EXEMPLO 2.18 - Contraposição:

$$p \rightarrow q \Leftrightarrow \neg q \rightarrow \neg p$$

Definição 2.15 - Prova por Contraposição
Uma prova é dita *Prova por Contraposição* ou *Demonstração por Contraposição* quando, para provar $p \rightarrow q$, prova-se $\neg q \rightarrow \neg p$, pois são formas equivalentes. Para provar $\neg q \rightarrow \neg p$ basta, a partir de $\neg q$, obter $\neg p$ (prova direta). ❑

EXEMPLO 2.27 - Prova por Contraposição
Para demonstrar o seguinte teorema (suponha $n \in \mathbf{N}$):

$$n! > (n+1) \rightarrow n > 2$$

pode-se, equivalentemente, demonstrar por contraposição que:

$$n \leq 2 \rightarrow n! \leq n+1$$

Observe que é muito simples provar que $n \leq 2 \rightarrow n! \leq n+1$, pois basta testar a proposição para os casos $n=0$, $n=1$ e $n=2$, o que se sugere como exercício. ❑

2.2.3 Prova por Redução ao Absurdo

A prova por redução ao absurdo baseia-se no seguinte resultado (denominado de *redução ao absurdo*), o qual foi verificado no EXEMPLO 2.19 - Redução ao Absurdo:

$$p \rightarrow q \Leftrightarrow (p \wedge \neg q) \rightarrow F$$

Definição 2.16 - Prova por Redução ao Absurdo
Uma prova é dita *Prova (Demonstração) por Redução ao Absurdo* ou simplesmente *Prova (Demonstração) por Absurdo* quando a prova de $p \rightarrow q$ consiste em supor a hipótese p, supor a negação da tese $\neg q$ e concluir uma contradição (em geral, $q \wedge \neg q$). ❑

Observe que a técnica de demonstração conhecida como *prova por contra-exemplo*, é uma demonstração por absurdo. De fato, em uma demonstração por absurdo, a construção da contradição $q \wedge \neg q$ é, em geral, a apresentação de um contra-exemplo.

EXEMPLO 2.28 - Prova por Redução ao Absurdo
Considere o seguinte teorema:

0 é o único elemento neutro da adição em **N**

ou seja, reescrevendo na forma de $p \rightarrow q$:

se 0 é elemento neutro da adição em **N**,
então 0 é o único elemento neutro da adição em **N**

Uma prova por redução ao absurdo é como segue:

a) Suponha que (hipótese) 0 é elemento neutro da adição em **N** e que (negação da tese) 0 *não* é o único elemento neutro da adição em **N**. Seja e um elemento neutro da adição em **N** tal que $e \neq 0$ (se 0 não é o único, então existe um outro, diferente de 0);

b) Então:
- como 0 é elemento neutro, para qualquer $n \in \mathbf{N}$, vale $n = 0 + n = n + 0$. Em particular, para $n = e$, vale $e = 0 + e = e + 0$
- como e é elemento neutro, para qualquer $n \in \mathbf{N}$, vale $n = n + e = e + n$. Em particular, para $n = 0$, vale $0 = 0 + e = e + 0$
- portanto, como $e = 0 + e = e + 0$ e $0 = 0 + e = e + 0$, pela transitividade da igualdade, vale $e = 0$, o que é uma contradição, pois foi suposto que $e \neq 0$

Logo, é absurdo supor que o elemento neutro da adição em **N** não é único. Portanto, 0 é o único elemento neutro da adição em **N**. ❑

2.3 Exercícios

Exercício 2.1 Sabendo que os valores-verdade das proposições p e q são respectivamente V e F, determine o valor lógico (V ou F) de cada uma das seguintes proposições:

a) $p \wedge \neg q$
b) $p \vee \neg q$
c) $\neg p \wedge q$
d) $\neg p \wedge \neg q$
e) $\neg p \vee \neg q$
f) $p \wedge (\neg p \vee q)$

Exercício 2.2 Determine $V(p)$ em cada um dos seguintes casos, sabendo que:

a) $V(q) = V$ e $V(p \wedge q) = F$
b) $V(q) = F$ e $V(p \vee q) = F$
c) $V(q) = F$ e $V(p \rightarrow q) = F$
d) $V(q) = F$ e $V(q \rightarrow p) = V$
e) $V(q) = V$ e $V(p \leftrightarrow q) = F$
f) $V(q) = F$ e $V(q \leftrightarrow p) = V$

Exercício 2.3 Determine $V(p)$ e $V(q)$ em cada um dos seguintes casos, sabendo que:

a) $V(p \rightarrow q) = V$ e $V(p \wedge q) = F$
b) $V(p \rightarrow q) = V$ e $V(p \vee q) = F$
c) $V(p \leftrightarrow q) = V$ e $V(p \wedge q) = V$
d) $V(p \leftrightarrow q) = V$ e $V(p \vee q) = V$
e) $V(p \leftrightarrow q) = F$ e $V(\neg p \vee q) = V$

Exercício 2.4 Construa as tabelas-verdade das seguintes fórmulas e identifique as que são tautologias ou contradições:

a) $\neg(p \vee \neg q)$
b) $\neg(p \rightarrow \neg q)$
c) $p \wedge q \rightarrow p \vee q$
d) $\neg p \rightarrow (q \rightarrow p)$
e) $(p \rightarrow q) \rightarrow p \wedge q$
f) $q \leftrightarrow \neg q \wedge p$
g) $p \rightarrow (q \rightarrow (q \rightarrow p))$

h) $\neg(p \rightarrow (\neg p \rightarrow q))$
i) $p \wedge q \rightarrow (p \leftrightarrow q \vee r)$
j) $\neg p \wedge r \rightarrow q \vee \neg r$
k) $p \rightarrow r \leftrightarrow q \vee \neg r$
l) $p \rightarrow (p \rightarrow \neg r) \leftrightarrow q \vee r$
m) $(p \wedge q \rightarrow r) \vee (\neg p \leftrightarrow q \vee \neg r)$

Exercício 2.5 Sabendo que as proposições $x = 0$ e $x = y$ são verdadeiras e que as proposições $y = z$ e $y = t$ são falsas, determinar o valor-verdade (V ou F) de cada uma das seguintes proposições:

a) $x = 0 \wedge x = y \rightarrow y \neq z$
b) $x \neq 0 \vee y = t \rightarrow y = z$
c) $x \neq y \vee y \neq z \rightarrow y = t$
d) $x \neq 0 \vee x \neq y \rightarrow y \neq z$
e) $x = 0 \rightarrow (x \neq y \vee y \neq t)$

Exercício 2.6 Considere a tabela-verdade ilustrada na Figura 2.11. Para cada uma das seguintes implicações, procure justificar os nomes associados:

a) *Adição.*

$p \Rightarrow p \vee q$

b) *Simplificação.*

$p \wedge q \Rightarrow p$

Exercício 2.7 Prove, usando tabela-verdade, as seguintes equivalências:

a) *Idempotência.*

$p \wedge p \Leftrightarrow p$
$p \vee p \Leftrightarrow p$

b) *Comutativa.*

$p \wedge q \Leftrightarrow q \wedge p$
$p \vee q \Leftrightarrow q \vee p$

c) *Associativa.*

$p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$
$p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$

d) *Distributiva.* A prova da distributividade do conetivo ou sobre o conetivo e, ou seja:

$p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$

foi apresentada na tabela-verdade ilustrada na Figura 2.8. Prove a distributividade do conetivo e se sobre o conetivo ou, ou seja:

$p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$

e) *Dupla negação.*

$\neg\neg p \Leftrightarrow p$

f) *DeMorgan* (Augustus DeMorgan, nascido na Índia, de família/educação inglesa, 1806-1871).

$\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$
$\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$

g) *Absorção.*

$p \wedge (p \vee q) \Leftrightarrow p$
$p \vee (p \wedge q) \Leftrightarrow p$

Exercício 2.8 Prove, usando tabela-verdade, as seguintes equivalências:

a) $p \leftrightarrow p \wedge q \Leftrightarrow p \rightarrow q$
b) $q \leftrightarrow p \vee q \Leftrightarrow p \rightarrow q$
c) $(p \rightarrow q) \wedge (p \rightarrow r) \Leftrightarrow p \rightarrow q \wedge r$
d) $(p \rightarrow q) \vee (p \rightarrow r) \Leftrightarrow p \rightarrow q \vee r$

Exercício 2.9 Prove, usando tabela-verdade, que qualquer dos conetivos estudados pode ser expresso usando somente os conetivos $\neg$ e $\wedge$.

Observação: este exercício é mais importante do que pode aparentar em um primeiro momento. De fato, esse resultado é usado ao longo do livro e é fundamental para estudos desenvolvidos em *Técnicas Digitais*, onde as diversas portas lógicas são expressas em termos de $\neg$ e $\wedge$.

Exercício 2.10 Prove, usando tabela-verdade, que os seguintes conetivos podem ser expressos usando os conetivos já estudados:

a) Conetivo EXOR (abreviatura dos termos em inglês *exclusive* e *or*) cuja semântica é dada pela tabela ilustrada na Figura 2.15;
b) Conetivo NAND (abreviatura dos termos em inglês *not* e *and*) cuja semântica é dada pela tabela ilustrada na Figura 2.15.

x	y	x EXOR y	x NAND y
V	V	F	F
V	F	V	V
F	V	V	V
F	F	F	V

Figura 2.15 Tabela-verdade: EXOR e NAND

Exercício 2.11 Suponha o conjunto universo **R**.

a) Determine o valor-verdade (V ou F) de cada uma das seguintes proposições:

a.1) $(\forall x)(|x| = x)$
a.2) $(\exists x)(x^2 = x)$
a.3) $(\exists x)(|x| = 0)$
a.4) $(\exists x)(x + 2 = x)$
a.5) $(\forall x)(x + 1 > x)$
a.6) $(\forall x)(x^2 = x)$
a.7) $(\exists x)(2x = x)$
a.8) $(\exists x)(x^2 + 3x = -2)$
a.9) $(\exists x)(x^2 + 5 = 2x)$
a.10) $(\forall x)(2x + 3x = 5x)$

b) Negue cada uma das proposições do item acima.

Exercício 2.12 Para ilustrar este exercício, considere o EXEMPLO 2.23 - Quantificador Universal, Quantificador Existencial. Observe que, no exemplo em questão, sempre que uma proposição quantificada universalmente é verdadeira, a mesma proposição, mas quantificada existencialmente, também é verdadeira. Esta observação vale para qualquer proposição (trocando o quantificador universal pelo existencial)? Justifique a sua resposta.

Exercício 2.13 Suponha o conjunto universo $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Apresente um contra-exemplo para cada uma das seguintes proposições:

a) $(\forall x)(x + 5 < 12)$
b) $(\forall x)(x \text{ é primo})$
c) $(\forall x)(x^2 > 1)$
d) $(\forall x)(x \text{ é par})$

Exercício 2.14 Suponha o conjunto universo $\{2, 3, 4, 5, 6, 7, 8, 9\}$.

a) Determine o valor-verdade (V ou F) de cada uma das seguintes proposições:

a.1) $(\forall x)(\forall y)(x + 5 < y + 12)$
a.2) $(\forall x)(\exists y)(x * y \text{ não é primo})$
a.3) $(\exists y)(\forall x)(x * y \text{ não é primo})$
a.4) $(\exists x)(\exists y)(x^2 > y)$
a.5) $(\forall x)(\exists y)(x^2 > y)$
a.6) $(\exists x)(\forall y)(x^2 > y)$
a.7) $(\forall x)(\forall y)(\forall z)(x + y > z)$
a.8) $(\exists x)(\forall y)(\forall z)(x + y > z)$
a.9) $(\forall x)(\exists y)(\forall z)(x + y > z)$
a.10) $(\forall x)(\forall y)(\exists z)(x + y > z)$
a.11) $(\forall x)(\exists y)(\exists z)(x + y > z)$
a.12) $(\exists x)(\forall y)(\exists z)(x + y > z)$
a.13) $(\exists x)(\exists y)(\forall z)(x + y > z)$

b) Negue cada uma das proposições do item acima.

Exercício 2.15 Para demonstrar o seguinte teorema:

$$n! > (n+1) \rightarrow n > 2$$

pode-se, equivalentemente, demonstrar por contraposição que:

$$n \leq 2 \rightarrow n! \leq n+1$$

Observe que é muito simples provar que $n \leq 2 \rightarrow n! \leq n+1$, pois é suficiente testar a proposição para os casos $n = 0$, $n = 1$ e $n = 2$. Detalhe esta prova.

3 Álgebra de Conjuntos

Álgebra, desde a sua origem até a sua forma atual, refere-se a cálculos. Com limitações, é desenvolvida de maneira informal ou formal, em praticamente todos os níveis de escolaridade. Por exemplo, as operações aritméticas básicas (adição, multiplicação etc.) sobre o conjunto dos números reais constituem uma álgebra.

Historicamente, o estudo das álgebras em Computação e Informática destaca-se a partir de 1950, com o desenvolvimento da Teoria dos Autômatos e Linguagens Formais. De certa forma, toda a Computação e Informática é baseada, direta ou indiretamente, sobre álgebras. Como comentado anteriormente, as Diretrizes Curriculares do MEC para Cursos de Computação e Informática [MEC 2005] referem-se à Álgebra como sendo uma denominação alternativa para a Matemática Discreta.

Um conceito mais formal de álgebra é introduzido ao longo do livro. Em um primeiro momento, considere (informalmente) que uma *Álgebra* é constituída de operações definidas sobre uma coleção de objetos. Nesse contexto, *Álgebra de Conjuntos* corresponderia às operações definidas sobre todos os conjuntos.

Antes de introduzir as operações sobre conjuntos, o seguinte é tratado:

a) *Diagramas de Venn*, os quais, usando uma representação diagramática, auxiliam o entendimento dos conceitos e raciocínios relacionados com conjuntos;
b) *Paradoxo de Russell*, o qual estabelece um importante resultado relativamente à Teoria dos Conjuntos.

As operações sobre conjuntos são classificadas em *reversíveis* e *não-reversíveis*. Embora as não-reversíveis sejam as mais usuais, as reversíveis são especialmente importantes para Computação e Informática, como será destacado adiante. As seguintes operações são definidas sobre conjuntos:

a) *Não-Reversíveis*.
- *União*;
- *Intersecção*;

b) *Reversíveis*.
- *Complemento*;
- *Conjunto das Partes*;
- *Produto Cartesiano*;
- *União Disjunta*.

Adicionalmente, é introduzida a operação de *diferença*, do tipo não-reversível, a qual é derivada a partir da composição das operações complemento e intersecção.

O estudo da Álgebra de Conjuntos fica sobremaneira facilitado considerando a seguinte observação.

Observação 3.1 - Lógica × Álgebra dos Conjuntos
No estudo da Álgebra de Conjuntos, o leitor atento poderá observar uma relação direta entre os conetivos lógicos introduzidos e as operações sobre conjuntos, como segue:

Conetivo Lógico	Operação sobre Conjuntos
negação	complemento
disjunção	união
conjunção	intersecção

Adicionalmente, as relações lógicas introduzidas também podem ser associadas com as relações sobre conjuntos, como segue:

Relação Lógica	Relação sobre Conjuntos
implicação	continência
equivalência	igualdade

Para reforçar esse relacionamento entre Lógica e Teoria dos Conjuntos, observe que a equivalência (respectivamente, igualdade de conjuntos) pode ser definida em termos de uma dupla implicação (respectivamente, dupla continência, ou seja, a "ida" e a "volta").

Da mesma forma, as propriedades introduzidas sobre os conetivos lógicos na Lógica são válidas na Teoria dos Conjuntos, substituindo cada conetivo pela correspondente operação sobre conjuntos, com destaque para as seguintes, as quais foram sugeridas como exercícios no Capítulo 2 - Lógica e Técnicas de Demonstração:

Conetivo Lógico	Operação sobre Conjuntos
idempotência: $\wedge$ e $\vee$	*idempotência*: intersecção e união
comutativa: $\wedge$ e $\vee$	*comutativa*: intersecção e união
associativa: $\wedge$ e $\vee$	*associativa*: intersecção e união
distributiva: $\wedge$ sobre $\vee$ $\vee$ sobre $\wedge$	*distributiva*: intersecção sobre união união sobre intersecção
dupla negação	*duplo complemento*
DeMorgan	*DeMorgan*
Absorção	*Absorção*

Baseando-se nesses fatos, é fácil intuir que as provas, na Teoria dos Conjuntos, são, em grande parte, baseadas em resultados da lógica. ❑

Excetuando-se o Paradoxo de Russell, provavelmente a grande maioria dos assuntos deste capítulo é do conhecimento do leitor, pois são desenvolvidos no Ensino Médio. Neste caso, sugere-se uma atenta passagem para verificar as nomenclaturas e convenções adotadas ao longo do livro, a relação com a Lógica, bem como a leitura da exemplificação de seu uso em Computação e Informática. Em particular, são introduzidos importantes tópicos da *Teoria da Computação*.

3.1 Diagramas de Venn

O tratamento dado aos conjuntos e conceitos correlatos até o momento usou uma linguagem textual. Entretanto, na medida em que outros conceitos são desenvolvidos, como as operações

sobre conjuntos, uma linguagem diagramática auxilia o entendimento de definições, facilita o desenvolvimento de raciocínios e permite uma identificação e uma compreensão fácil e rápida dos componentes e dos relacionamentos em discussão.

Os *Diagramas de Venn* (John Venn (1834-1923), matemático inglês) são universalmente conhecidos e são largamente usados nos estudos da Teoria dos Conjuntos. Os diagramas usam figuras geométricas, em geral representadas no plano, para expressar as estruturas da Teoria dos Conjuntos. O conceito é introduzido via exemplos. Assim, na Figura 3.1, são ilustradas as seguintes construções (da esquerda para a direita):

um dado conjunto A
um determinado elemento $b \in B$
o conjunto $C = \{1, 2, 3\}$

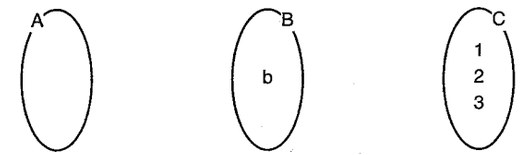


Figura 3.1 Diagramas de Venn

Da mesma forma, na Figura 3.2, são ilustradas as seguintes construções (da esquerda para a direita):

$\{a, b\} \subseteq \{a, b, c\}$
$A \subseteq B$
para um dado conjunto universo $\mathbf{U}$, um conjunto $C \subseteq \mathbf{U}$.

As figuras usadas em um Diagrama de Venn podem ser diversas. Em geral, o conjunto universo é representado por um retângulo e os demais conjuntos por círculos, elipses, etc. Observe que, no caso $C \subseteq \mathbf{U}$, o conjunto C é destacado, para auxiliar visualmente o que se deseja representar. Tal tipo de destaque é importante na interpretação das operações sobre conjuntos introduzidas adiante.

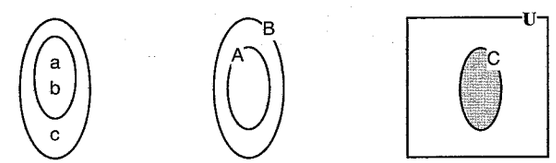


Figura 3.2 Diagramas de Venn

Para ilustrar uma aplicação dos Diagramas de Venn, considere a Figura 3.3. Claramente, pode-se intuir que a noção de subconjunto satisfaz a propriedade de transitividade, ou seja, para quaisquer conjuntos A, B e C, vale:

$$A \subseteq B \wedge B \subseteq C \Rightarrow A \subseteq C$$

De fato, tal implicação pode ser comprovada pelo seguinte teorema.

Teorema 3.2 - Transitividade da Continência
Suponha que A, B e C são conjuntos quaisquer. Se $A \subseteq B$ e $B \subseteq C$, então $A \subseteq C$

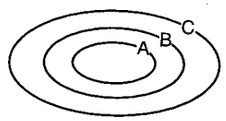


Figura 3.3 Transitividade da continência

Prova: (*direta*)

Lembre-se de que $X \subseteq Y$ se e somente se todos os elementos de X também são elementos de Y.

Suponha que A, B e C são conjuntos quaisquer e que $A \subseteq B$ e $B \subseteq C$

Seja $a \in A$. Então (na coluna da direita é apresentada uma breve justificativa da implicação):

$a \in A \Rightarrow$	pela definição de subconjunto, considerando que $A \subseteq B$
$a \in B \Rightarrow$	pela definição de subconjunto, considerando que $B \subseteq C$
$a \in C$	

Portanto, para qualquer elemento $a \in A$, vale $a \in C$

Logo, pela definição de subconjunto, vale $A \subseteq C$ (como fica a demonstração se A for vazio? Observe que, neste caso, não existe elemento $a \in A$). ❑

3.2 Paradoxo de Russell

Lembre-se de que a definição de conjunto é:

uma coleção de zero ou mais elementos distintos
os quais não possuem qualquer ordem associada

Considerando que conjuntos podem possuir conjuntos como elementos, surge a seguinte dúvida:

um conjunto pode ser elemento de si mesmo?

Com o objetivo de excluir essa dúvida, introduz-se a noção de *conjunto ordinário*, ou seja, um conjunto que não pertence a si mesmo. Nesse contexto, considere a seguinte definição por compreensão:

$$S = \{A \mid A \text{ é um conjunto ordinário}\}$$

ou seja:

o conjunto de todos os conjuntos que não são elementos de si mesmos

Entretanto, como é verificada a seguir, tal definição determina uma contradição, denominada de *Paradoxo de Russell* (enunciada pelo matemático e filósofo Bertrand Russell (1872-1970) em 1901).

Teorema 3.3 - Paradoxo de Russell

A seguinte definição *não* é um conjunto:

$$S = \{A \mid A \text{ é um conjunto ordinário}\}$$

Prova: (*por absurdo*)

Lembre-se de que uma demonstração por absurdo consiste em supor a hipótese, supor a negação da tese e concluir uma contradição.

Negação da tese. Suponha que S *é* um conjunto.

Construção da contradição. Como S é um conjunto de conjuntos, uma questão natural é verificar se S é um elemento de si mesmo. Assim:

Caso 1. Suponha que $S \in S$. Então (na coluna da direita é apresentada uma breve justificativa da implicação):

$S \in S \Rightarrow$	pela definição de conjunto ordinário
S não é um conjunto ordinário $\Rightarrow$	pela definição de S
$S \notin S$	

Caso 2. Suponha que $S \notin S$. Então (na coluna da direita é apresentada uma breve justificativa da implicação):

$S \notin S \Rightarrow$	pela definição de conjunto ordinário
S é um conjunto ordinário $\Rightarrow$	pela definição de S
$S \in S$	

Claramente, em qualquer dos dois casos, existe uma contradição.

Logo, é absurdo supor que S é um conjunto; portanto, S *não* é um conjunto. ❑

Portanto, a notação por compreensão permite definir algo que não é um conjunto. Adicionalmente, observe que S seria um subconjunto do conjunto de todos os conjuntos. Como, pelo Paradoxo de Russell, S não é conjunto, então se pode afirmar que:

não existe o conjunto de todos os conjuntos

ou seja:

nem toda coleção de elementos constitui um conjunto

Uma forma simples de evitar esse paradoxo (quando o objetivo é definir um conjunto por compreensão) é restringir que a, em $\{a \mid p(a)\}$, assuma valores em um determinado conjunto A, ou seja, sempre especificar na forma:

$$\{a \in A \mid p(a)\}$$

O Paradoxo de Russell possui uma importante conseqüência na definição de uma estrutura matemática sobre uma coleção de elementos. Convenciona-se denominar tal estrutura de:

- *pequena*, se a coleção de elementos considerada *é* um conjunto;
- *grande*, se a coleção de elementos considerada *não* é um conjunto.

Portanto, a Álgebra de Conjuntos é uma *álgebra grande*, pois é constituída de operações definidas sobre a coleção (a qual não é um conjunto) de todos os conjuntos.

Quando se deseja trabalhar sobre uma *álgebra pequena*, é usual considerar um dado conjunto universo **U**. Nesse caso, qualquer operando A é tal que $A \subseteq \mathbf{U}$. Esse tipo de álgebra é introduzido adiante, quando do estudo da operação conjunto das partes.

3.3 Operações Não-Reversíveis

As operações não-reversíveis são as mais comuns nos estudos usualmente desenvolvidos sobre Álgebra de Conjuntos.

3.3.1 União

Considere o diagrama ilustrado na Figura 3.10. A operação de união é uma operação binária, a qual, quando aplicada a dois conjuntos A e B, resulta em um conjunto constituído pelos elementos que pertencem a pelo menos um dos dois conjuntos, ou seja, que pertencem ao conjunto A ou ao conjunto B.

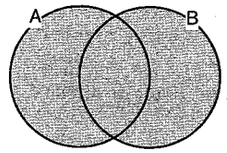


Figura 3.4 União

Definição 3.4 - União, Reunião
Sejam A e B conjuntos. A *União* ou *Reunião* dos conjuntos A e B, denotada por:

$$A \cup B$$

é como segue:

$$A \cup B = \{x \mid x \in A \vee x \in B\}$$ ❑

Relacionando com a Lógica, a união corresponde à noção de disjunção. Ou seja, $A \cup B$ considera todos os elementos que pertencem ao conjunto A *ou* ao conjunto B. Observe que o símbolo de união $\cup$ lembra símbolo de disjunção $\vee$.

EXEMPLO 3.1 - União

a) Suponha os conjuntos:

$$\text{Dígitos} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$
$$\text{Vogais} = \{a, e, i, o, u\}$$
$$\text{Pares} = \{0, 2, 4, 6, \ldots\}$$

Considere a Figura 3.5. Então:

$$\text{Dígitos} \cup \text{Vogais} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, e, i, o, u\}$$
$$\text{Dígitos} \cup \text{Pares} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, \ldots\}$$

b) Suponha os conjuntos $A = \{x \in \mathbf{N} \mid x > 2\}$ e $B = \{x \in \mathbf{N} \mid x^2 = x\}$. Então:

$$A \cup B = \{0, 1, 3, 4, 5, 6, \ldots\}$$

c) Considere os conjuntos **R** (reais), **Q** (racionais) e **I** (irracionais). Então:

$$\mathbf{R} \cup \mathbf{Q} = \mathbf{R}$$
$$\mathbf{R} \cup \mathbf{I} = \mathbf{R}$$
$$\mathbf{Q} \cup \mathbf{I} = \mathbf{R}$$

d) Para qualquer conjunto universo **U** e qualquer $A \subseteq \mathbf{U}$, vale:

$$\varnothing \cup \varnothing = \varnothing$$
$$\mathbf{U} \cup \varnothing = \mathbf{U}$$
$$\mathbf{U} \cup A = \mathbf{U}$$
$$\mathbf{U} \cup \mathbf{U} = \mathbf{U}$$

❑

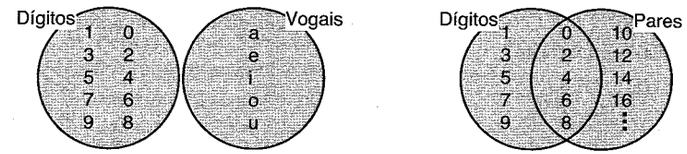


Figura 3.5 União

As seguintes propriedades da operação união podem ser facilmente verificadas (suponha os conjuntos A, B e C):

a) *Elemento Neutro.*

$$A \cup \varnothing = \varnothing \cup A = A$$

b) *Idempotência.*

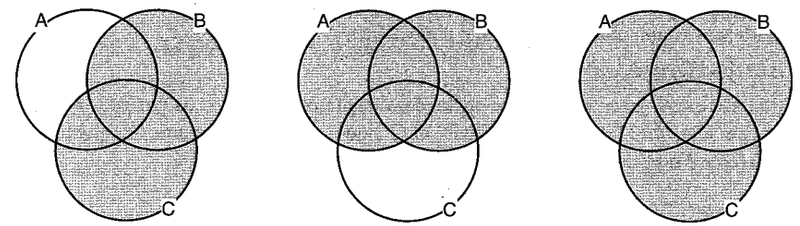
$$A \cup A = A$$

c) *Comutativa.*

$$A \cup B = B \cup A$$

d) *Associativa.* Na Figura 3.6 é ilustrada a associatividade usando Diagramas de Venn.

$$A \cup (B \cup C) = (A \cup B) \cup C$$



$B \cup C$ $\quad$ $A \cup B$ $\quad$ $A \cup (B \cup C) = (A \cup B) \cup C$

Figura 3.6 Associatividade da união

A prova de que, de fato, as propriedades elemento neutro, idempotência e comutativa são válidas para a união são sugeridas como exercício. Segue a prova da associatividade. Observe que, para demonstrar a associatividade da união, é suposta a associatividade da disjunção. Essa prova ilustra o fato já comentado de que cada propriedade da disjunção corresponde a uma propriedade da união e vice-versa (portanto, qual o elemento neutro da disjunção?).

Teorema 3.5 - Associatividade da União
Suponha que A, B e C são conjuntos quaisquer. Então:

$$A \cup (B \cup C) = (A \cup B) \cup C$$

<u>Prova:</u> (*direta*)
Lembre-se de que, para dois conjuntos X e Y, para provar que $X = Y$, pode-se, equivalentemente, provar que $X \subseteq Y$ e $Y \subseteq X$.

Suponha que A, B e C são conjuntos quaisquer. A prova é dividida em dois casos:

$A \cup (B \cup C) \subseteq (A \cup B) \cup C$ *caso 1*
$(A \cup B) \cup C \subseteq A \cup (B \cup C)$ *caso 2*

Caso 1. Suponha $x \in A \cup (B \cup C)$. Então (na coluna da direita é apresentada uma breve justificativa da implicação):

$x \in A \cup (B \cup C) \Rightarrow$	
$x \in A \vee x \in (B \cup C) \Rightarrow$	definição de união
$x \in A \vee (x \in B \vee x \in C) \Rightarrow$	definição de união
$(x \in A \vee x \in B) \vee x \in C \Rightarrow$	associatividade do conetivo $\vee$
$x \in (A \cup B) \vee x \in C \Rightarrow$	definição de união
$x \in (A \cup B) \cup C$	definição de união

Portanto, $A \cup (B \cup C) \subseteq (A \cup B) \cup C$

Caso 2. Suponha $x \in (A \cup B) \cup C$. Então (na coluna da direita é apresentada uma breve justificativa da implicação):

$x \in (A \cup B) \cup C \Rightarrow$	
$x \in (A \cup B) \vee x \in C \Rightarrow$	definição de união
$(x \in A \vee x \in B) \vee x \in C \Rightarrow$	definição de união
$x \in A \vee (x \in B \vee x \in C) \Rightarrow$	associatividade do conetivo $\vee$
$x \in A \vee x \in (B \cup C) \Rightarrow$	definição de união
$x \in A \cup (B \cup C)$	definição de união

Portanto, $(A \cup B) \cup C \subseteq A \cup (B \cup C)$

Observe que o *caso 1* e o *caso 2* são análogos, trocando o sentido da implicação. Seria possível reduzir essa prova a um único caso, usando equivalências?

Logo, $A \cup (B \cup C) = (A \cup B) \cup C$ ❑

Assim, não existe precedência entre operações de união e, portanto, os parênteses podem ser omitidos, ou seja, $A \cup (B \cup C)$ ou $(A \cup B) \cup C$ pode, equivalentemente, ser denotado sem parênteses (esse é o significado da propriedade associativa) como segue:

$$A \cup B \cup C$$

3.3.2 Intersecção

Considere o diagrama ilustrado na Figura 3.7. A operação de intersecção é uma operação binária a qual, quando aplicada a dois conjuntos A e B, resulta em um conjunto constituído pelos elementos que pertencem simultaneamente aos dois conjuntos, ou seja, que pertencem ao conjunto A e ao conjunto B.

Definição 3.6 - Intersecção

Sejam A e B conjuntos. A *Intersecção* dos conjuntos A e B, denotada por:

$$A \cap B$$

é como segue:

$$A \cap B = \{x \mid x \in A \wedge x \in B\}$$ ❑

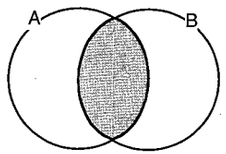


Figura 3.7 Intersecção

Relacionando com a Lógica, a intersecção corresponde à noção de conjunção. Ou seja, $A \cap B$ considera todos os elementos que pertencem ao conjunto A *e* ao conjunto B. Observe que o símbolo de intersecção $\cap$ lembra o símbolo de conjunção $\wedge$.

Dados dois conjuntos A e B, sendo ambos não-vazios, se $A \cap B = \varnothing$, então A e B são ditos *conjuntos disjuntos*, *conjuntos independentes* ou *conjuntos mutuamente exclusivos*.

EXEMPLO 3.2 - Intersecção, Conjuntos Disjuntos

a) Suponha os conjuntos:

$\text{Dígitos} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$\text{Vogais} = \{a, e, i, o, u\}$
$\text{Pares} = \{0, 2, 4, 6, \ldots\}$

Considere a Figura 3.8. Então:

$\text{Dígitos} \cap \text{Vogais} = \varnothing$ conjuntos disjuntos
$\text{Dígitos} \cap \text{Pares} = \{0, 2, 4, 6, 8\}$

b) Suponha os conjuntos $A = \{x \in \mathbf{N} \mid x > 2\}$ e $B = \{x \in \mathbf{N} \mid x^2 = x\}$. Então:

$A \cap B = \varnothing$ conjuntos disjuntos

c) Considere os conjuntos **R** (reais), **Q** (racionais) e **I** (irracionais). Então:

$\mathbf{R} \cap \mathbf{Q} = \mathbf{Q}$
$\mathbf{R} \cap \mathbf{I} = \mathbf{I}$
$\mathbf{Q} \cap \mathbf{I} = \varnothing$ conjuntos disjuntos

d) Para qualquer conjunto universo **U** e qualquer $A \subseteq \mathbf{U}$, vale:

$\varnothing \cap \varnothing = \varnothing$ por que o vazio não é disjunto de si próprio?
$\mathbf{U} \cap \varnothing = \varnothing$
$\mathbf{U} \cap A = A$
$\mathbf{U} \cap \mathbf{U} = \mathbf{U}$ ❑

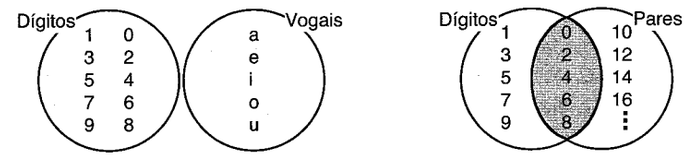


Figura 3.8 Intersecção

As seguintes propriedades da operação intersecção podem ser facilmente verificadas (suponha o conjunto universo **U** e os conjuntos A, B e C):

a) *Elemento Neutro.*

$A \cap \mathbf{U} = \mathbf{U} \cap A = A$

b) *Idempotência.*

$A \cap A = A$

c) *Comutativa.*

$A \cap B = B \cap A$

d) *Associativa.*

$A \cap (B \cap C) = (A \cap B) \cap C$

A prova de que, de fato, as propriedades acima são válidas para a intersecção são sugeridas como exercício.

Adicionalmente às propriedades da união e da intersecção já apresentadas, existem importantes propriedades que envolvem as duas operações, como segue (suponha os conjuntos A, B e C):

a) *Distributividade da intersecção sobre a união.* A propriedade é ilustrada na Figura 3.9, usando Diagramas de Venn;

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

b) *Distributividade da união sobre a intersecção.*

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

c) *Absorção.*

$$A \cap (A \cup B) = A$$
$$A \cup (A \cap B) = A$$

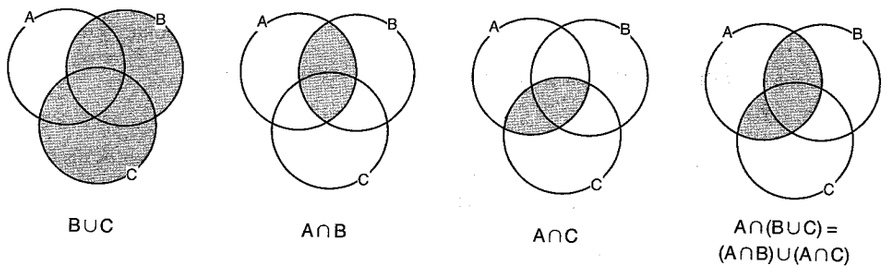


Figura 3.9 Distributividade da intersecção sobre a união

Segue a prova da distributividade da intersecção sobre a união. Observe que, para demonstrar esta propriedade, é suposta a distributividade da conjunção sobre a disjunção. Esta prova ilustra, mais uma vez, a correlação entre as propriedades da Lógica e a Teoria dos Conjuntos e vice-versa (portanto, qual o elemento neutro da conjunção?).

Teorema 3.7 - Distributividade da intersecção sobre a união

Suponha que A, B e C são conjuntos quaisquer. Então:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

Prova: (*direta*)

Suponha que A, B e C são conjuntos quaisquer. Então (na coluna da direita é apresentada uma breve justificativa da equivalência):

$x \in A \cap (B \cup C) \Leftrightarrow$	definição de intersecção
$x \in A \wedge x \in (B \cup C) \Leftrightarrow$	definição de união
$x \in A \wedge (x \in B \vee x \in C) \Leftrightarrow$	distributividade do $\wedge$ sobre o $\vee$
$(x \in A \wedge x \in B) \vee (x \in A \wedge x \in C) \Leftrightarrow$	definição de intersecção
$x \in (A \cap B) \vee x \in (A \cap C) \Leftrightarrow$	definição de união
$x \in (A \cap B) \cup (A \cap C)$	

Compare com a prova do Teorema 3.5 - Associatividade da União. Observe que essa não foi dividida em dois casos. Por quê?

Logo $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ ❑

3.4 Operações Reversíveis

Entende-se por operação reversível uma operação a partir de cujo resultado pode-se recuperar os operandos originais. A reversibilidade de uma operação é importante em muitas aplicações na Computação e Informática. Para efeitos ilustrativos, considere os seguintes casos:

a) *Back Tracking.* Entende-se por *back tracking* como o processo de recuperação dos operandos originais a partir do resultado da operação realizada. Por exemplo, considere uma simples operação de débito e crédito em um terminal bancário informatizado. Claramente, é uma operação resultante da composição de diversas pequenas operações componentes. Suponha que ocorre uma queda de sistema (como a ocasionada por uma falta de luz) entre duas operações componentes. Nesse caso, o sistema poderia ficar inconsistente (por exemplo, o débito foi realizado, mas o correspondente crédito, não). Portanto, quando o sistema volta a operar, é fundamental desfazer o que foi parcialmente feito (no caso, o débito). Obviamente, a recuperação fica facilitada se a operação a ser recuperada for reversível;

b) *Construção de Estruturas Complexas.* Estruturas computacionais complexas são usualmente construídas compondo estruturas elementares já conhecidas. Na maioria dos casos, é desejável que qualquer alteração realizada em alguma estrutura elementar seja refletida na estrutura composta. Tal reflexão só é possível se forem conhecidos os elementos originais da estrutura resultante. Obviamente, a recuperação dessa informação fica facilitada se as operações usadas para construir a estrutura complexa forem reversíveis.

De fato, existem diversas outras aplicações em Computação e Informática para as quais a reversibilidade de uma operação é importante.

3.4.1 Complemento

Considere os diagramas ilustrados na Figura 3.10. A operação de complemento é uma operação unária a qual é definida em relação a um conjunto universo **U**. Assim, a operação de complemento, aplicada a um conjunto A (diagrama à esquerda), resulta, como o próprio nome indica, no conjunto complemento do conjunto A em relação ao **U**, ou seja, a região do diagrama à direita destacada e identificada por ~A.

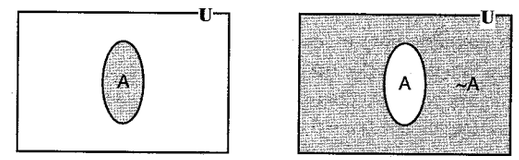


Figura 3.10 Um conjunto (esquerda) e o seu complemento (direita)

Definição 3.8 - Complemento

Suponha o conjunto universo **U**. O *Complemento* de um conjunto $A \subseteq \mathbf{U}$, denotado por:

$$A' \quad \text{ou} \quad \sim A$$

é como segue:

$$\sim A = \{x \in \mathbf{U} \mid x \notin A\}$$ ❑

Relacionando com a Lógica, o complemento corresponde à noção de negação. Ou seja, considera todos os elementos do universo que *não* pertencem ao conjunto original. Observe que o símbolo de complemento ~ é um dos símbolos usados para a negação na Lógica.

EXEMPLO 3.3 - Complemento

a) Considere a Figura 3.11. Suponha o conjunto universo $\text{Dígitos} = \{0, 1, 2, \ldots, 9\}$. Seja $A = \{0, 1, 2\}$. Então:

$$\sim A = \{3, 4, 5, 6, 7, 8, 9\}$$

b) Suponha o conjunto universo **N**. Seja $A = \{0, 1, 2\}$. Então:

$$\sim A = \{x \in \mathbf{N} \mid x > 2\}$$

c) Para qualquer conjunto universo **U**, vale:

$$\sim\varnothing = \mathbf{U}$$
$$\sim\mathbf{U} = \varnothing$$

d) Suponha o conjunto **R** como conjunto universo. Então:

$$\sim\mathbf{Q} = \mathbf{I}$$
$$\sim\mathbf{I} = \mathbf{Q}$$

❑

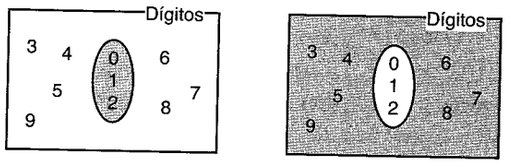


Figura 3.11 Um conjunto (esquerda) e o seu complemento (direita)

EXEMPLO 3.4 - Complemento, União e Intersecção

Suponha que **U** é o conjunto universo. Então, para qualquer conjunto $A \subseteq \mathbf{U}$, vale (por quê?):

$$A \cup \sim A = \mathbf{U}$$
$$A \cap \sim A = \varnothing$$

Observe que:

- a união de um conjunto com o seu complemento sempre resulta no conjunto universo (compare com o fato de que $p \vee \neg p$ é uma tautologia);
- a intersecção de um conjunto com o seu complemento sempre resulta no conjunto vazio (compare com o fato de que $p \wedge \neg p$ é uma contradição). ❑

Uma importante propriedade da operação de complemento, decorrente da sua própria definição, denominada de *duplo complemento*, é o fato de que, para um dado conjunto $A \subseteq \mathbf{U}$, vale:

$$\sim\sim A = A$$

Essa conclusão pode ser facilmente intuída simplesmente observando os diagramas na Figura 3.10. Observe que tal conclusão está diretamente relacionada com a dupla negação da Lógica. De fato:

- para um conjunto A, considera-se todos os elementos x tais que $x \in A$;
- para o ~A, considera-se todos os elementos x tais que $x \notin A$, ou seja, tais que $\neg(x \in A)$
- conseqüentemente, para ~~A, considera-se todos os elementos x tais que $\neg\neg(x \in A)$, ou seja, tais que $x \in A$

Considerando a propriedade da dupla negação, conclui-se que a operação de complemento é reversível. De fato, a partir do resultado ~A, é possível recuperar o operando original aplicando a própria operação de complemento ao resultado, ou seja, $\sim(\sim A) = A$

Existe uma outra importante propriedade, denominada de *DeMorgan*, relacionada com a operação de complemento, e que envolve as operações de união e de intersecção, como abaixo (na coluna da direita é apresentada a correspondente propriedade da Lógica) e ilustrada na Figura 3.12:

$$\sim(A \cup B) = \sim A \cap \sim B \qquad \neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$$
$$\sim(A \cap B) = \sim A \cup \sim B \qquad \neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$$

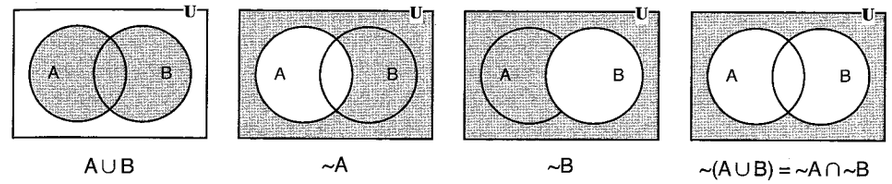


Figura 3.12 DeMorgan: complemento da união é a intersecção dos complementos

Essa propriedade permite concluir que a intersecção (respectivamente, a união) pode ser calculada em termos das operações de complemento e de união (respectivamente, de intersecção), ou seja, que (por quê?):

$$A \cap B = \sim(\sim A \cup \sim B)$$
$$A \cup B = \sim(\sim A \cap \sim B)$$

Considere o diagrama ilustrado na Figura 3.13. Uma operação derivada das operações já estudadas é a diferença a qual, quando aplicada a dois conjuntos A e B, resulta em um conjunto constituído pelos elementos que pertencem ao conjunto A e não pertencem ao conjunto B.

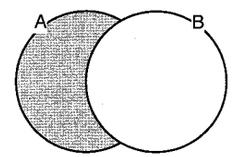


Figura 3.13 Diferença: A - B

Definição 3.9 - Diferença

Sejam A e B conjuntos. A *Diferença* dos conjuntos A e B, denotada por:

$$A - B$$

é como segue:

$$A - B = A \cap \sim B$$

ou seja:

$$A - B = \{x \mid x \in A \wedge x \notin B\}$$ ❑

EXEMPLO 3.5 - Diferença

a) Suponha os conjuntos:

$\text{Dígitos} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
$\text{Vogais} = \{a, e, i, o, u\}$
$\text{Pares} = \{0, 2, 4, 6, \ldots\}$

Considere a Figura 3.14. Então:

$\text{Dígitos} - \text{Vogais} = \text{Dígitos}$
$\text{Dígitos} - \text{Pares} = \{1, 3, 5, 7, 9\}$

b) Suponha os conjuntos $A = \{x \in \mathbf{N} \mid x > 2\}$ e $B = \{x \in \mathbf{N} \mid x^2 = x\}$. Então:

$A - B = \{3, 4, 5, 6, \ldots\}$
$B - A = \{0, 1\}$

c) Considere os conjuntos **R** (reais), **Q** (racionais) e **I** (irracionais). Então:

$\mathbf{R} - \mathbf{Q} = \mathbf{I}$
$\mathbf{R} - \mathbf{I} = \mathbf{Q}$
$\mathbf{Q} - \mathbf{I} = \mathbf{Q}$

d) Para qualquer conjunto universo **U** e qualquer $A \subseteq \mathbf{U}$, vale:

$\varnothing - \varnothing = \varnothing$
$\mathbf{U} - \varnothing = \mathbf{U}$
$\mathbf{U} - A = \sim A$
$\mathbf{U} - \mathbf{U} = \varnothing$ ❑

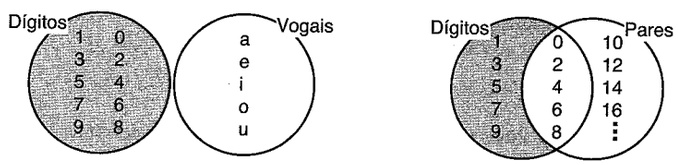


Figura 3.14 Diferença: Dígitos - Vogais (esquerda) e Dígitos - Pares (direita)

Por que a operação de diferença é não-reversível?

3.4.2 Conjunto das Partes

Para um dado conjunto A, sabe-se que:

$A \subseteq A$
$\varnothing \subseteq A$

Adicionalmente, para qualquer elemento $a \in A$ é claro que:

$\{a\} \subseteq A$

Seguindo esse raciocínio, pode-se definir uma operação unária, denominada de conjunto das partes a qual, quando aplicada a um conjunto A, resulta no conjunto de todos os subconjuntos de A.

Definição 3.10 - Conjunto das Partes, Conjunto Potência

Suponha um conjunto A. O *Conjunto das Partes* de A ou *Conjunto Potência* de A, denotado por:

$$\mathbf{P}(A) \quad \text{ou} \quad 2^A$$

é como segue:

$$\mathbf{P}(A) = \{X \mid X \subseteq A\}$$ ❑

EXEMPLO 3.6 - Conjunto das Partes

Sejam $A = \{a\}$, $B = \{a, b\}$, $C = \{a, b, c\}$ e $D = \{a, \varnothing, \{a, b\}\}$. Então:

$\mathbf{P}(\varnothing) = \{\varnothing\}$
$\mathbf{P}(A) = \{\varnothing, \{a\}\}$
$\mathbf{P}(B) = \{\varnothing, \{a\}, \{b\}, \{a, b\}\}$
$\mathbf{P}(C) = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$

Sugere-se especial atenção ao seguinte caso:

$\mathbf{P}(D) = \{\varnothing, \{a\}, \{\varnothing\}, \{\{a, b\}\}, \{a, \varnothing\}, \{a, \{a, b\}\}, \{\varnothing, \{a, b\}\}, \{a, \varnothing, \{a, b\}\}\}$ ❑

Supondo que o número de elementos de X é n, pode-se provar que o número de elementos de $\mathbf{P}(X)$ é 2^n (comprove tal fato no EXEMPLO 3.6 - Conjunto das Partes), justificando a notação alternativa 2^X. A prova desse resultado é, em geral, realizada usando a técnica de demonstração denominada *prova por indução*, a qual será introduzida ao longo deste livro.

Como, a partir do resultado da operação $\mathbf{P}(X)$, pode-se recuperar o operando original? Uma forma (existem outras) é realizar a união de todos os elementos (conjuntos) de $\mathbf{P}(X)$. Uma questão delicada que não será discutida ao longo deste livro é como ficaria o cálculo dessa união se o número de elementos do conjunto das partes for infinito (resultando em uma composição infinita de uniões).

EXEMPLO 3.7 - Reversibilidade do Conjunto das Partes

Para cada um dos itens abaixo, é apresentado o conjunto resultante de um conjunto das partes e os correspondentes operandos originais:

a) Resultante: $\{\varnothing, \{a\}\}$
Operando: $\varnothing \cup \{a\} = \{a\}$

b) Resultante: $\{\varnothing, \{a\}, \{b\}, \{a, b\}\}$
Operando: $\varnothing \cup \{a\} \cup \{b\} \cup \{a, b\} = \{a, b\}$

c) Resultante: $\{\varnothing, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\}$
Operando: $\varnothing \cup \{a\} \cup \{b\} \cup \{c\} \cup \{a, b\} \cup \{a, c\} \cup \{b, c\} \cup \{a, b, c\} = \{a, b, c\}$ ❑

Observação 3.11 - Álgebra de Conjuntos Pequena

Já foi introduzido que a Álgebra de Conjuntos é uma álgebra grande, pois é constituída de operações definidas sobre a coleção (a qual não é um conjunto) de todos os conjuntos. Entretanto, quando é desejado trabalhar sobre uma Álgebra de Conjuntos pequena, basta considerar um dado conjunto **U** e definir a álgebra em questão sobre $\mathbf{P}(\mathbf{U})$, originando, para cada **U** considerado, uma álgebra diferente.

Nesse caso, qualquer operando A é tal que $A \in \mathbf{P}(\mathbf{U})$. Portanto, operações como união, intersecção, diferença e complemento (em relação ao **U**) são *operações fechadas* sobre $\mathbf{P}(\mathbf{U})$, ou

seja, são operações tais que, para quaisquer operandos de **P(U)**, o resultado também é um elemento de **P(U)**.

Entretanto, a operação de conjunto das partes não necessariamente é fechada sobre **P(U)** (por quê?). ❑

3.4.3 Produto Cartesiano

A operação produto cartesiano é uma operação binária a qual, quando aplicada a dois conjuntos A e B, resulta em um conjunto constituído de seqüências de duas componentes, sendo que a primeira componente de cada seqüência é um elemento de A, e a segunda componente, um elemento de B.

Assim, para definir produto cartesiano, é necessário antes introduzir a noção de *seqüência finita* e, em particular, de seqüência de dois elementos. Uma seqüência de n componentes, denominada de n-*upla ordenada* consiste de n objetos (não necessariamente distintos) em uma ordem fixa. Em particular, uma 2-upla ordenada é denominada de *par ordenado*. Um par ordenado no qual a primeira componente é x e a segunda é y é denotado como segue:

$$\langle x, y\rangle \quad \text{ou} \quad (x, y)$$

Analogamente, uma n-upla ordenada é denotada como segue:

$$\langle x_1, x_2, x_3, \ldots, x_n\rangle \quad \text{ou} \quad (x_1, x_2, x_3, \ldots, x_n)$$

Uma n-upla ordenada $\langle x_1, x_2, x_3, \ldots, x_n\rangle$ não deve ser confundida com o conjunto $\{x_1, x_2, x_3, \ldots, x_n\}$: em uma n-upla ordenada, a ordem é importante, pois são distinguidas as componentes. Por exemplo, para x e y distintos:

$$\langle x, y\rangle \neq \langle y, x\rangle$$

Definição 3.12 - Produto Cartesiano

Sejam A e B conjuntos. O *Produto Cartesiano* dos conjuntos A e B, denotado por:

$$A \times B$$

é como segue:

$$A \times B = \{\langle a, b\rangle \mid a \in A \text{ e } b \in B\}$$ ❑

É usual denotar um produto cartesiano de um conjunto com ele mesmo como um expoente. Por exemplo:

$$A \times A = A^2$$

EXEMPLO 3.8 - Produto Cartesiano

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então:

$A \times B = \{\langle a, a\rangle, \langle a, b\rangle\}$
$B \times C = \{\langle a, 0\rangle, \langle a, 1\rangle, \langle a, 2\rangle, \langle b, 0\rangle, \langle b, 1\rangle, \langle b, 2\rangle\}$
$C \times B = \{\langle 0, a\rangle, \langle 0, b\rangle, \langle 1, a\rangle, \langle 1, b\rangle, \langle 2, a\rangle, \langle 2, b\rangle\}$
$A^2 = \{\langle a, a\rangle\}$
$B^2 = \{\langle a, a\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle b, b\rangle\}$
$A \times \mathbf{N} = \{\langle a, 0\rangle, \langle a, 1\rangle, \langle a, 2\rangle, \langle a, 3\rangle, \ldots\}$
$(A \times B) \times C = \{\langle\langle a, a\rangle, 0\rangle, \langle\langle a, a\rangle, 1\rangle, \langle\langle a, a\rangle, 2\rangle, \langle\langle a, b\rangle, 0\rangle, \langle\langle a, b\rangle, 1\rangle, \langle\langle a, b\rangle, 2\rangle\}$
$A \times (B \times C) = \{\langle a, \langle a, 0\rangle\rangle, \langle a, \langle a, 1\rangle\rangle, \langle a, \langle a, 2\rangle\rangle, \langle a, \langle b, 0\rangle\rangle, \langle a, \langle b, 1\rangle\rangle, \langle a, \langle b, 2\rangle\rangle\}$ ❑

Relativamente ao EXEMPLO 3.8 - Produto Cartesiano, observe que:

a) *Não-Comutativa*. Os conjuntos resultantes $B \times C$ e $C \times B$ são *diferentes*. De fato, neste caso, são disjuntos, ou seja, $(B \times C) \cap (C \times B) = \varnothing$. Portanto, a operação produto cartesiano é *não-comutativa*;

b) *Não-Associativa*. Analogamente, $(A \times B) \times C$ e $A \times (B \times C)$ são conjuntos *diferentes*, pois as componentes de cada par ordenado são distintas. Logo, a operação produto cartesiano é *não-associativa*.

EXEMPLO 3.9 - Produto Cartesiano

Seja $A = \{0, 1, 2\}$. Então (por quê?):

$A \times \varnothing = \varnothing$
$\varnothing \times A = \varnothing$
$\varnothing^2 = \varnothing$ ❑

O produto cartesiano se distribui sobre a união e sobre a intersecção. De fato, sugere-se provar como exercício que (suponha os conjuntos A, B e C):

a) *Distributividade do produto cartesiano sobre a união.*

$A \times (B \cup C) = (A \times B) \cup (A \times C)$

b) *Distributividade do produto cartesiano sobre a intersecção.*

$A \times (B \cap C) = (A \times B) \cap (A \times C)$

A reversibilidade do produto cartesiano pode ser facilmente obtida como segue: o primeiro operando (respectivamente, o segundo operando) é o conjunto constituído por todos os elementos da primeira (respectivamente, da segunda) componente dos pares do produto cartesiano. Entretanto, a reversibilidade nem sempre é válida quando o produto cartesiano resulta no conjunto vazio (por quê? *Dica*: observe o EXEMPLO 3.9 - Produto Cartesiano).

EXEMPLO 3.10 - Reversibilidade do Produto Cartesiano

Para cada um dos itens abaixo, é apresentado o conjunto resultante de um produto cartesiano e os correspondentes operandos originais:

a) Resultante: $\{\langle a, a\rangle, \langle a, b\rangle\}$
Operandos: $\{a\}$ e $\{a, b\}$

b) Resultante: $\{\langle a, a\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle b, b\rangle\}$
Operandos: $\{a, b\}$ e $\{a, b\}$

c) Resultante: $\{\langle a, 0\rangle, \langle a, 1\rangle, \langle a, 2\rangle, \langle a, 3\rangle, \ldots\}$
Operandos: $\{a\}$ e $\mathbf{N}$

d) Resultante: $\{\langle\langle a, a\rangle, 0\rangle, \langle\langle a, a\rangle, 1\rangle, \langle\langle a, a\rangle, 2\rangle, \langle\langle a, b\rangle, 0\rangle, \langle\langle a, b\rangle, 1\rangle, \langle\langle a, b\rangle, 2\rangle\}$
Operandos: $\{\langle a, a\rangle, \langle a, b\rangle\}$ e $\{0, 1, 2\}$ ❑

3.4.4 União Disjunta

Suponha os conjuntos A e B. A operação de união $A \cup B$ considera que elementos com mesma identificação nos dois conjuntos (por exemplo, $x \in A$ e $x \in B$) sejam considerados uma única vez no conjunto resultante, ou seja, existe somente um elemento x em $A \cup B$. Entretanto, considere os seguintes conjuntos que representam pessoas da família Silva e Souza:

Silva = {João, Maria, José}
Souza = {Pedro, Ana, José}

Obviamente, no conjunto resultante da união dos conjuntos das famílias, José ocorre uma única vez, ou seja:

$$\text{Silva} \cup \text{Souza} = \{\text{João, Maria, Pedro, Ana, José}\}$$

Entretanto, tal situação não reflete a realidade correspondente a uma "reunião familiar", pois José da família Silva não é o mesmo José da família Souza.

Com o objetivo de distinguir elementos com uma mesma identificação, é definida a união disjunta, ou seja, um tipo de união no qual é garantido que não existem elementos em comum. Na definição de união disjunta que segue, a técnica usada para garantir a não-existência de elementos comuns é associar uma identificação do conjunto origem, constituindo um tipo de "sobrenome". Assim, os elementos do conjunto resultante da união disjunta são pares da forma:

$$\langle \text{elemento, identificação do conjunto origem} \rangle$$

Portanto, no conjunto resultante da união disjunta das duas famílias, José aparece *duas* vezes, mas cada um com um "sobrenome" identificando se trata-se do José da família Silva ou o José da família Souza.

Definição 3.13 - União Disjunta

Sejam A e B conjuntos. A *União Disjunta* dos conjuntos A e B, denotada por:

$$A + B \quad \text{ou} \quad A \uplus B$$

é como segue:

$$A + B = \{\langle a, A\rangle \mid a \in A\} \cup \{\langle b, B\rangle \mid b \in B\}$$

Ou, alternativamente:

$$A + B = \{\langle a, 0\rangle \mid a \in A\} \cup \{\langle b, 1\rangle \mid b \in B\}$$
$$A + B = \{a_A \mid a \in A\} \cup \{b_B \mid b \in B\}$$

❑

Existem diversas formas alternativas de denotar os elementos de $A + B$. O importante é que a forma convencionada deixe claro de qual operando o elemento do conjunto resultante é originário.

EXEMPLO 3.11 - União Disjunta

a) Suponha os conjuntos $\text{Silva} = \{\text{João, Maria, José}\}$ e $\text{Souza} = \{\text{Pedro, Ana, José}\}$. Então:

$$\text{Silva} + \text{Souza} = \{\langle \text{João, Silva}\rangle, \langle \text{Maria, Silva}\rangle, \langle \text{José, Silva}\rangle, \langle \text{Pedro, Souza}\rangle, \langle \text{Ana, Souza}\rangle, \langle \text{José, Souza}\rangle\}$$

b) Suponha os conjuntos

$$D = \{0, 1, 2, \ldots, 9\}$$
$$V = \{a, e, i, o, u\}$$
$$P = \{0, 2, 4, 6, \ldots\}$$

Então:

$$D + V = \{0_D, 1_D, 2_D, \ldots, 9_D, a_V, e_V, i_V, o_V, u_V\}$$
$$D + P = \{0_D, 1_D, 2_D, \ldots, 9_D, 0_P, 2_P, 4_P, 6_P, \ldots\}$$

c) Suponha os conjuntos $A = \{x \in \mathbf{N} \mid x > 2\}$ e $B = \{x \in \mathbf{N} \mid x^2 = x\}$. Então:

$$A + B = \{0_B, 1_B, 3_A, 4_A, 5_A, 6_A, \ldots\}$$

d) Considere o conjunto $A = \{a, b, c\}$. Então:

$$\varnothing + \varnothing = \varnothing$$
$$A + \varnothing = \{\langle a, A\rangle, \langle b, A\rangle, \langle c, A\rangle\}$$
$$A + A = \{\langle a, 0\rangle, \langle b, 0\rangle, \langle c, 0\rangle, \langle a, 1\rangle, \langle b, 1\rangle, \langle c, 1\rangle\}$$

❑

A união disjunta é uma operação reversível. De fato, como cada elemento possui uma identificação de qual conjunto é originário, então os operandos podem ser reconstruídos.

EXEMPLO 3.12 - Reversibilidade da União Disjunta

Para cada um dos itens abaixo, é apresentado o conjunto resultante de uma união disjunta e os correspondentes operandos originais:

a) Resultante: $\{0_D, 1_D, 2_D, \ldots, 9_D, a_V, e_V, i_V, o_V, u_V\}$
Operandos: $\{0, 1, 2, \ldots, 9\}$ e $\{a, e, i, o, u\}$

b) Resultante: $\{0_D, 1_D, 2_D, \ldots, 9_D, 0_N, 1_N, 2_N, 3_N \ldots\}$
Operandos: $\{0, 1, 2, \ldots, 9\}$ e $\mathbf{N}$

c) Resultante: $\varnothing$
Operandos: $\varnothing$ e $\varnothing$

d) Resultante: $\{\langle a, 0\rangle, \langle b, 0\rangle\}$
Operandos: $\{a, b\}$ e $\varnothing$

e) Resultante: $\{\langle a, 0\rangle, \langle b, 0\rangle, \langle a, 1\rangle, \langle b, 1\rangle, \langle c, 1\rangle\}$
Operandos: $\{a, b\}$ e $\{a, b, c\}$

❑

3.5 Relação entre Lógica e Álgebra de Conjuntos

Como já destacado na Observação 3.1 - Lógica × Álgebra dos Conjuntos, existe uma relação direta entre os conetivos lógicos e algumas operações sobre conjuntos. A tabela na Figura 3.15 resume alguns dos principais resultados discutidos.

Um importante exercício proposto no Capítulo 2 - Lógica e Técnicas de Demonstração é que qualquer dos conetivos estudados ($\neg$, $\wedge$, $\vee$, $\rightarrow$ e $\leftrightarrow$) pode ser expresso usando somente os conetivos $\neg$ e $\wedge$. Esse resultado é importante em diversas aplicações da Computação e Informática como, por exemplo, nos estudos das *Técnicas Digitais*. Conseqüentemente, o mesmo resultado vale para a Álgebra de Conjuntos, usando somente as operações $\sim$ e $\cap$, respectivamente. Considerando a solução apresentada para este exercício, como os conetivos $\rightarrow$ e $\leftrightarrow$ podem ser correspondentemente expressos na Álgebra de Conjuntos?

Adicionalmente, existe uma analogia entre as relações lógicas e as relações sobre conjuntos. A tabela na Figura 3.16 resume os resultados discutidos.

No Capítulo 1 - Introdução e Conceitos Básicos foi introduzida informalmente a definição de um conjunto por extensão da forma:

$$A = \{x \mid p(x)\}$$

Conforme visto no Capítulo 2 - Lógica e Técnicas de Demonstração, $p(x)$ é uma proposição p a qual descreve alguma propriedade de um elemento $x \in \mathbf{U}$. Assim, reforçando a relação entre Lógica a Teoria dos Conjuntos, a continência e a igualdade de dois conjuntos $A = \{x \mid p(x)\}$ e $B = \{x \mid q(x)\}$ pode ser vista como segue:

$A \subseteq B$ se e somente se $(\forall x \in \mathbf{U})\,(p(x) \Rightarrow q(x))$ *continência*
$A = B$ se e somente se $(\forall x \in \mathbf{U})\,(p(x) \Leftrightarrow q(x))$ *igualdade*

Propriedade	Lógica	Teoria dos Conjuntos
Idempotência	$p \wedge p \Leftrightarrow p$ $p \vee p \Leftrightarrow p$	$A \cap A = A$ $A \cup A = A$
Comutativa	$p \wedge q \Leftrightarrow q \wedge p$ $p \vee q \Leftrightarrow q \vee p$	$A \cap B = B \cap A$ $A \cup B = B \cup A$
Associativa	$p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$ $p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$	$A \cap (B \cap C) = (A \cap B) \cap C$ $A \cup (B \cup C) = (A \cup B) \cup C$
Distributiva	$p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$ $p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$	$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$ $A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$
Negação/ Complemento	$\neg\neg p \Leftrightarrow p$ $p \wedge \neg p \Leftrightarrow F$ $p \vee \neg p \Leftrightarrow V$	$\sim\sim A = A$ $A \cap \sim A = \varnothing$ $A \cup \sim A = \mathbf{U}$
DeMorgan	$\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$ $\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$	$\sim(A \cup B) = \sim A \cap \sim B$ $\sim(A \cap B) = \sim A \cup \sim B$
Elemento Neutro	$p \wedge V \Leftrightarrow p$ $p \vee F \Leftrightarrow p$	$A \cap \mathbf{U} = A$ $A \cup \varnothing = A$
Elemento Absorvente	$p \wedge F \Leftrightarrow F$ $p \vee V \Leftrightarrow V$	$A \cap \varnothing = \varnothing$ $A \cup \mathbf{U} = \mathbf{U}$
Absorção	$p \wedge (p \vee q) \Leftrightarrow p$ $p \vee (p \wedge q) \Leftrightarrow p$	$A \cap (A \cup B) = A$ $A \cup (A \cap B) = A$

Figura 3.15 Conetivos lógicos × operações sobre conjuntos

Relação	Lógica	Teoria dos Conjuntos
Implicação/Continência	$p \Rightarrow q$	$A \subseteq B$
Equivalência/Igualdade	$p \Leftrightarrow q$	$A = B$

Figura 3.16 Relações lógicas × relações sobre conjuntos

Assim, por exemplo:

$A = \mathbf{U}$ se e somente se $(\forall x \in \mathbf{U})\ (p(x) \Leftrightarrow V)$ *universo*
$A = \varnothing$ se e somente se $(\forall x \in \mathbf{U})\ (p(x) \Leftrightarrow F)$ *vazio*

Esse raciocínio justifica o fato de que qualquer continência (respectivamente, igualdade) na Teoria dos Conjuntos é decorrência de alguma implicação (respectivamente, equivalência) de correspondentes fórmulas lógicas.

Essa correlação direta entre Lógica e Álgebra de Conjuntos não é casual. De fato, ambas são um caso particular de uma álgebra abstrata denominada *Álgebra de Boole* ou *Álgebra Booleana*, a qual será vista adiante.

3.6 Álgebra de Conjuntos nas Linguagens de Programação

Como já discutido anteriormente, nem toda linguagem de programação possui boas facilidades para tratar conjuntos. No caso específico da linguagem Pascal, lembre-se de que é possível definir tipos de dados baseados em conjuntos finitos, variáveis conjuntos sobre esses tipos de dados, bem como constantes conjuntos (também finitos). Pascal possui as seguintes operações não-reversíveis sobre conjuntos:

+ (união)
* (intersecção)
- (diferença)

EXEMPLO 3.13 - Trechos de Programas em Pascal

Suponha que é definido o seguinte tipo de dados:

```
dias_semana = set of (seg, ter, qua, qui, sex, sab, dom)
```

bem como as seguintes variáveis:

```
feriado, trabalho, feriado_trabalho, úteis, parados: dias_semana
```

Considere, também, os seguintes trechos de programas:

```
feriado := [qua, sab]
```

```
trabalho := [seg,..., sex]
```

Assim, os seguintes trechos de programas em Pascal:

```
feriado_trabalho := trabalho * feriado
```

```
úteis := trabalho - feriado
```

```
parado := [sab, dom] + feriado
```

correspondem, na Teoria dos Conjuntos, aos seguintes conjuntos:

$\text{feriado_trabalho} = \text{trabalho} \cap \text{feriado}$ (conjunto resultante: $\{\text{qua}\}$)
$\text{úteis} = \text{trabalho} - \text{feriado}$ (conjunto resultante: $\{\text{seg, ter, qui, sex}\}$)
$\text{parado} = \{\text{sab, dom}\} \cup \text{feriado}$ (conjunto resultante: $\{\text{qua, sab, dom}\}$)

❑

EXEMPLO 3.14 - Programa em Pascal

O objetivo deste exemplo é introduzir um programa completo em Pascal, de fácil entendimento, o qual usa algumas das construções introduzidas sobre conjuntos.

Assim, suponha que se deseja desenvolver um programa em Pascal capaz de ler uma linha de texto e determinar o número de vogais, de consoantes e de outros símbolos, bem como o número total de caracteres lidos.

Um programa é apresentado na Figura 3.17. Observe o seguinte:

- o nome do programa é `numero_caracteres` e usa os procedimentos predefinidos do sistema para entradas (`input`) e saídas (`output`);
- após a palavra `type`, é definido o tipo de dado (sobre conjuntos) `alfabeto`;
- após a palavra `var`, são definidas as variáveis do tipo `inteiro`, `alfabeto` e `char` (tipo de dado referente aos caracteres);

```
program numero_caracteres(input, output);
type
      alfabeto = set of 'a'..'z';
var
      num_vogais, num_consoantes, num_outros, total: integer;
      vogais, consoantes: alfabeto;
      caractere: char;
begin
      vogais := ['a', 'e', 'i', 'o', 'u'];
      consoantes := ['a'..'z'] - vogais;
      num_vogais := 0;
      num_consoantes := 0;
      num_outros := 0;
      read(caractere);
      while not eoln
      do    begin
            if    caractere in vogais
            then  num_vogais := num_vogais + 1
            else  if    caractere in consoantes
                  then  num_consoantes := num_consoantes + 1
                  else  num_outros := num_outros + 1;
            read(caractere)
            end;
      total := num_vogais + num_consoantes + num_outros;
      writeln('vogais = ', num_vogais);
      writeln('consoantes: ', num_consoantes);
      writeln('outros símbolos: ', num_outros);
      writeln('total de símbolos: ', total)
end.
```

Figura 3.17 Programa em Pascal

- entre as palavras `begin` e o correspondente `end` são especificados os comandos do programa (definem as ações);
- a variável `vogais` é inicializada com o conjunto de todas as vogais;
- a variável `consoantes` é inicializada com o conjunto de todas as consoantes, resultante de uma diferença entre conjuntos;
- as variáveis `num_vogais`, `num_consoantes` e `num_outros` são inicializadas com o valor inteiro zero;
- o comando `read(caractere)` lê o próximo caractere;
- o comando `while-do` tem a seguinte semântica: enquanto a expressão lógica após a palavra `while` for verdadeira, o comando após a palavra `do` é executado repetidamente;
- portanto, o comando `while-do` fica executando enquanto o fim da linha não é atingido (ou seja, enquanto não há mais caracteres na linha para ser lido). A palavra `eoln` é uma abreviatura da expressão em inglês *end of line*;
- no comando `while-do`, após a palavra `do`, são especificados dois comandos: `if-then-else` (de fato, são dois comandos `if-then-else` encadeados), introduzido anteriormente, e `read`. Para evitar possíveis ambigüidades, esses comandos são agrupados em um bloco delimitado pelas palavras `begin-end`. Observe que, se o agrupamento for omitido, não é claro se o comando `read` está dentro ou fora do escopo do `while-do`;
- os dois comandos `if-then-else` encadeados, no escopo do `while-do`, identificam se o caractere lido é vogal, consoante ou outro símbolo e atualizam o correspondente contador;
- após terminar o ciclo de repetição referente ao `while-do`, calcula o total de caracteres lidos e imprime os valores calculados. A palavra `writeln` é uma abreviatura da expressão em inglês *write line*. ❑

Adicionalmente às operações não-reversíveis de união, de intersecção e de diferença, a linguagem Pascal possui, de forma predefinida, construções similares à do produto cartesiano as quais são reversíveis. Outras operações, como conjunto das partes e união disjunta, *não* são predefinidas (embora possam ser definidas pelo programador).

As construções inspiradas no produto cartesiano são as seguintes:

- arranjos (em inglês, *arrays*);
- registros (em inglês, *records*).

Entretanto, uma abordagem mais adequada para essas construções pode ser desenvolvida quando do estudo do conceito de *função*.

3.7 Álgebra de Conjuntos e Teoria da Computação

Álgebra de Conjuntos é fundamental no estudo da *Teoria da Computação* a qual, resumidamente, fornece meios para uma correta aplicação e entendimento dos conceitos de algoritmo, de computabilidade e, conseqüentemente, do que é solucionável em um sistema computador. De certa forma, Teoria da Computação refere-se aos conceitos mínimos que qualquer estudante de Computação e Informática necessita saber.

O que segue desenvolve uma breve introdução dos limites do que é computável, ou seja, do que é possível resolver em um sistema computador, e estabelece algumas conclusões correlatas baseadas na Álgebra de Conjuntos.

Lembre-se de que uma linguagem (formal) L sobre um alfabeto (conjunto finito) Σ é um subconjunto de Σ^*, ou seja:

$$L \subseteq \Sigma^*$$

Assim, o complemento da linguagem L é:

$$\sim L = \{x \in \Sigma^* \mid x \notin L\}$$

EXEMPLO 3.15 - Complemento de Linguagens

Suponha o alfabeto $\Sigma = \{a, b\}$ e as seguintes linguagens:

$L_1 = \{\varepsilon\}$

$L_2 = \{a\}^* = \{\varepsilon, a, aa, aaa, \ldots\}$

Palíndromos $= \{\varepsilon, a, b, aa, bb, aaa, aba, bab, bbb, aaaa, \ldots\}$

Então:

$\sim L_1 = \{a, b, aa, ab, ba, bb, aaa, \ldots\}$

$\sim L_2 = \{b, ab, ba, bb, aab, aba, baa, abb, bab, bba, bbb, \ldots\}$

$$\sim\text{Palíndromos} = \{x \in \Sigma^* \mid x \notin \text{Palíndromos}\}$$ ❑

A observação que segue pode parecer óbvia e desnecessária, considerando o que já foi introduzido sobre compiladores. Entretanto, deve ser lida com atenção, pois tem importante implicação no estudo de toda a Computação e Informática, como destacado na observação subseqüente.

Observação 3.14 - Reconhecimento de Linguagens × Complemento

Lembre-se de que um compilador é um *software* que traduz um programa escrito na linguagem de programação para um código executável no sistema computador. Já foi comentado que o compilador normalmente é estruturado em análise e síntese. A análise é a estrutura responsável pelo *reconhecimento da linguagem*. Assim, verifica se um dado programa p é, de fato, um programa válido para a linguagem L em questão, ou seja, verifica se:

$$p \in L$$

Nesse caso, o compilador pode passar para a fase de síntese. Caso contrário ($p \notin L$), o compilador deve alertar o programador para que este corrija os eventuais problemas do programa. Portanto, um compilador também deve ser capaz de afirmar se $p \notin L$, o que significa verificar se $p \in \sim L$. Logo, a análise de um compilador verifica se o programa fornecido de fato pertence à linguagem ou ao complemento da linguagem, ou seja, verifica se:

$$p \in L \quad \text{ou} \quad p \in \sim L$$ ❑

Observação 3.15 - Hierarquia de Linguagens e Problema da Parada

Um importante assunto estudado em *Teoria da Computação* trata dos limites do que é possível computar em um sistema computador. No Capítulo 2 - Lógica e Técnicas de Demonstração, foi dito que a *Máquina de Turing* (introduzida no Capítulo 7 - Cardinalidade de Conjuntos) é aceita como uma formalização do conceito de algoritmo computável. Referente aos limites do que é possível reconhecer (existe uma Máquina de Turing capaz de reconhecer), alguns importantes resultados são provados, os quais possuem influência direta em toda a Computação e Informática. Nesse contexto, as linguagens são agrupadas em classes, determinando a hierarquia (continência própria) de classes de linguagens ilustrada na Figura 3.18, como segue (suponha que L denota uma linguagem sobre um alfabeto Σ):

a) *Linguagens Recursivas*. São linguagens para as quais *existe* um algoritmo que *sempre pára*, capaz de determinar se uma determinada palavra p pertence ou não à linguagem, ou seja, para uma dada linguagem L, existe uma Máquina de Turing MT(L) que determina se $p \in L$ ou $p \in \sim L$, como ilustrado na Figura 3.19 (esquerda);

b) *Linguagens Recursivamente Enumeráveis*. Considere a Figura 3.19 (direita). São linguagens para as quais *existe* um algoritmo capaz de determinar se uma determinada palavra p pertence à linguagem, ou seja, se $p \in L$. Entretanto, se $p \in \sim L$, o algoritmo pode:
   - *parar*, identificando que não pertence à linguagem;
   - *ficar em loop* infinito (processando indefinidamente).

   Esse resultado contradiz a intuição da maioria das pessoas, pois estabelece que:

   reconhecer o complemento de uma linguagem pode ser impossível, mesmo que seja possível reconhecer a linguagem

c) *Linguagens Não-Computáveis*. Como o próprio nome indica, são linguagens para as quais *não* existe um algoritmo capaz de determinar se uma determinada palavra p pertence ou não à linguagem, ou seja, determinar se $p \in L$ ou $p \in \sim L$;

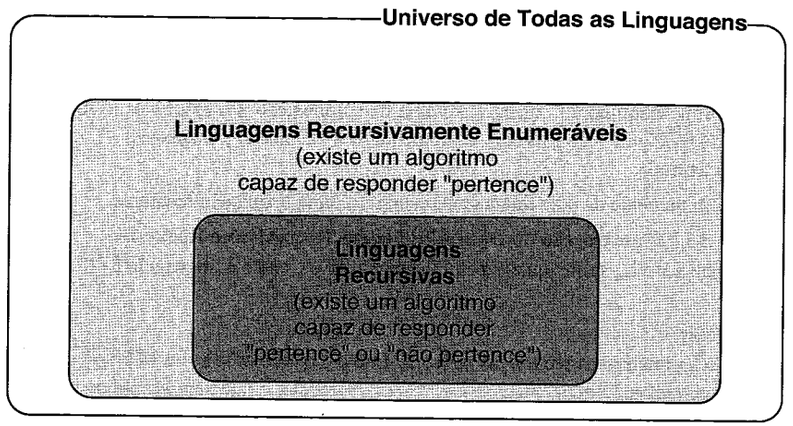


Figura 3.18 Hierarquia (continência própria) de classes de linguagens

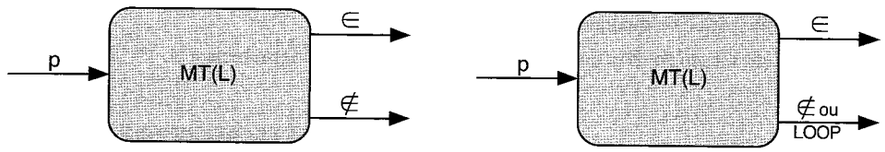


Figura 3.19 Linguagem recursiva (esquerda) e recursivamente enumerável (direita)

O problema que objetiva determinar se uma Máquina de Turing qualquer pára, determinando se $p \in L$ ou $p \in \sim L$, é conhecido como o *Problema da Parada* o qual *não tem solução* computacional. Trata-se de um resultado fundamental, pois, baseado nesse, prova-se que inúmeros outros problemas também não possuem solução computacional. Infelizmente, muitos dos problemas interessantes e importantes para a Computação e Informática, bem como para as ciências em geral, são não-solucionáveis computacionalmente. ❑

Nesse contexto, os seguintes resultados podem ser facilmente verificados:

a) O complemento de uma linguagem recursiva é uma linguagem recursiva;
b) A intersecção de duas linguagens recursivas é uma linguagem recursiva;
c) A união de duas linguagens recursivas é uma linguagem recursiva;
d) O complemento de uma linguagem recursivamente enumerável não necessariamente é uma linguagem recursivamente enumerável;
e) Uma linguagem é recursiva se e somente se a linguagem e seu complemento são linguagens recursivamente enumeráveis.

Para ilustrar como tais resultados podem ser verificados, os dois primeiros são apresentados a seguir.

Teorema 3.16 - Complemento de uma Linguagem Recursiva é Recursiva

Se uma linguagem L sobre um alfabeto Σ é recursiva, então a linguagem ~L também é recursiva.

Prova: (*direta*)

Suponha L uma linguagem recursiva sobre Σ. Então, existe MT(L), Máquina de Turing, que aceita a linguagem e sempre pára para qualquer entrada.

Seja Inverte uma Máquina de Turing que, para a entrada pertence (respectivamente, não-pertence), retorna a saída não-pertence (respectivamente, pertence), como ilustrado na Figura 3.20 (esquerda).

Seja MT'(L) uma Máquina de Turing resultante da composição das máquinas Inverte e MT(L), como ilustrado na Figura 3.20 (direita). Claramente, MT'(L) aceita a linguagem ~L e sempre pára para qualquer entrada.

Portanto, o complemento de uma linguagem recursiva é uma linguagem recursiva. ❑

Teorema 3.17 - Intersecção de Duas Linguagens Recursivas é Recursiva
Se as linguagens L_1 e L_2 sobre um alfabeto Σ são recursivas, então a linguagem $L_1 \cap L_2$ também é recursiva.

Prova: (*direta*)
Suponha L_1 e L_2 linguagens recursivas sobre Σ. Então existem $MT(L_1)$ e $MT(L_2)$, Máquinas de Turing, que aceitam as linguagens L_1 e L_2, respectivamente, e que sempre param para qualquer entrada.

Seja $MT(L_1 \cap L_2)$ uma Máquina de Turing resultante da composição das máquinas $MT(L_1)$ e $MT(L_2)$, como ilustrado na Figura 3.21. Portanto, $MT(L_1 \cap L_2)$ aceita a linguagem $L_1 \cap L_2$ (por quê?) e sempre pára para qualquer entrada.

Portanto, a intersecção de duas linguagens recursivas é uma linguagem recursiva. ❑

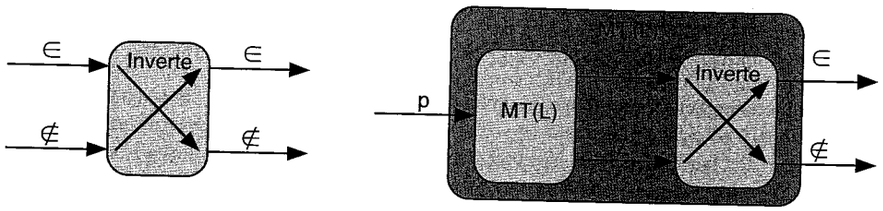


Figura 3.20 Complemento de uma linguagem recursiva é recursiva

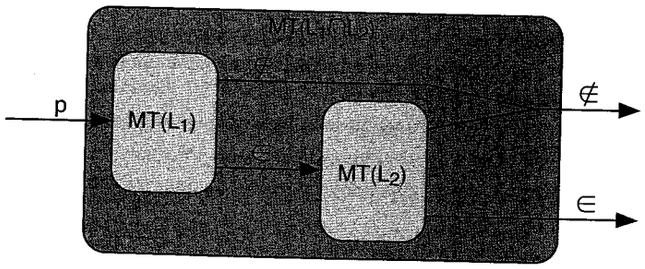


Figura 3.21 Intersecção de duas linguagens recursivas é recursiva

3.8 Exercícios

Exercício 3.1 Considere o Teorema 3.2 - Transitividade da Continência. No caso em que $A \subseteq B$ e $B \subseteq C$, como fica a demonstração se A for vazio? Observe que, neste caso, não existe elemento $a \in A$.

Exercício 3.2 Suponha o conjunto universo $S = \{p, q, r, s, t, u, v, w\}$ bem como os seguintes conjuntos:

$A = \{p, q, r, s\}$
$B = \{r, t, v\}$
$C = \{p, s, t, u\}$

Então, determine:

a) $B \cap C$
b) $A \cup C$
c) $\sim C$
d) $A \cap B \cap C$
e) $B - C$
f) $\sim(A \cup B)$
g) $A \times B$
h) $(A \cup B) \cap \sim C$
i) $A + B$
j) $B + B$

Exercício 3.3 Suponha o conjunto universo $S = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ bem como os seguintes conjuntos:

$A = \{2, 4, 5, 6, 8\}$
$B = \{1, 4, 5, 9\}$
$C = \{x \mid x \in \mathbf{Z} \wedge 2 \leq x < 5\}$

Então, determine:

a) $A \cup B$
b) $A \cap B$
c) $A \cap C$
d) $B \cup C$
e) $A - B$
f) $\sim A$
g) $A \cap \sim A$
h) $\sim(A \cap B)$
i) $C - B$
j) $(C \cap B) \cup \sim A$
k) $\sim(B - A) \cap (A - B)$
l) $\sim(\sim C \cup B)$
m) $B \times C$
n) $(A \times B) \times C$
o) $B + C$

p) $(A+B)+C$
q) $(B+B)+B$

Exercício 3.4 Prove as seguintes propriedades da operação de união (suponha A e B conjuntos):

a) *Elemento Neutro.*

$$A \cup \varnothing = \varnothing \cup A = A$$

b) *Idempotência.*

$$A \cup A = A$$

c) *Comutativa.*

$$A \cup B = B \cup A$$

Exercício 3.5 Considere o Teorema 3.5 - Associatividade da União. Observe que o *caso 1* e o *caso 2* são análogos, trocando o sentido da implicação. Seria possível reduzir essa prova a um único caso, usando equivalências? Nesse caso, como ficaria a prova?

Exercício 3.6 Prove as seguintes propriedades da operação de intersecção (suponha o conjunto universo **U** bem como quaisquer conjuntos A, B e C):

a) *Elemento Neutro.*

$$A \cap \mathbf{U} = \mathbf{U} \cap A = A$$

b) *Idempotência.*

$$A \cap A = A$$

c) *Comutativa.*

$$A \cap B = B \cap A$$

d) *Associativa.*

$$A \cap (B \cap C) = (A \cap B) \cap C$$

Exercício 3.7 Relativamente à união e à intersecção, prove as seguintes propriedades:

a) *Distributividade da união sobre a intersecção*, ou seja (suponha A, B e C conjuntos quaisquer):

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

b) *Absorção*, ou seja (suponha A e B conjuntos quaisquer):

$$A \cap (A \cup B) = A$$
$$A \cup (A \cap B) = A$$

Exercício 3.8 Considere a propriedade de *DeMorgan*, relacionada com a operação de complemento e que envolve as operações de união e de intersecção. Prove que a intersecção (respectivamente, a união) pode ser calculada em termos das operações de complemento e de união (respectivamente, de intersecção), ou seja, que:

a) $A \cap B = \sim(\sim A \cup \sim B)$
b) $A \cup B = \sim(\sim A \cap \sim B)$

Exercício 3.9 Para uma dada operação binária $\oplus$ sobre um conjunto A, afirma-se que a operação $\oplus$ possui *elemento absorvente* se existe $a \in A$ tal que, para qualquer $x \in A$ vale:

$$a \oplus x = x \oplus a = a$$

Mostre que as seguintes operações possuem elemento absorvente:

a) União;
b) Intersecção;
c) Produto cartesiano.

Por que as seguintes operações não possuem elemento absorvente? Justifique:

d) Diferença;
e) União Disjunta.

Exercício 3.10 Prove que (suponha A, B e C conjuntos quaisquer):

a) $(A \cup B) \cap \sim A = B \cap \sim A$
b) $(A \cap B) \cup A = A$
c) $A \cup (\sim A \cap B) = A \cup B$
d) $A \cap (\sim A \cup B) = A \cap B$
e) $\sim((A \cap B) \cup (\sim A \cap \sim B)) = (\sim A \cap B) \cup (A \cap \sim B)$

Exercício 3.11 Prove que (suponha A, B e C conjuntos quaisquer):

a) $A - B \subseteq A$
b) $A - B = A \Leftrightarrow A \cap B = \varnothing$
c) $(A - B) \cap B = \varnothing$
d) $(A - B) \cup B = A \cup B$
e) $A \cap B = A - (A - B)$
f) $A - \sim B = \varnothing \Leftrightarrow A \cap B = \varnothing$
g) $A - (B \cap C) = (A - B) \cup (A - C)$
h) $A - (B \cup C) = (A - B) \cap (A - C) = (A - B) - C$

Exercício 3.12 Por que a operação de diferença é não-reversível?

Exercício 3.13 Verifique se a operação de diferença *satisfaz* ou *não satisfaz* as seguintes propriedades (suponha A, B e C conjuntos quaisquer):

a) Elemento Neutro, ou seja, se existe algum conjunto E tal que:

$$A - E = E - A = A$$

b) Idempotência.

$$A - A = A$$

c) Comutativa.

$$A - B = B - A$$

d) Associativa.

$$A - (B - C) = (A - B) - C$$

Exercício 3.14 Verifique se a operação de união disjunta *satisfaz* ou *não satisfaz* às seguintes propriedades (suponha A, B e C conjuntos quaisquer):

a) Elemento Neutro, ou seja, se existe algum conjunto E tal que:

$$A + E = E + A = A$$

b) Idempotência.

$$A + A = A$$

c) Comutativa.

$$A + B = B + A$$

d) Associativa.

$$A + (B + C) = (A + B) + C$$

Exercício 3.15 Para um dado conjunto S, considere o conjunto universo como sendo **P**(S). Mostre que a operação conjunto das partes não necessariamente é fechada sobre **P**(S).

Exercício 3.16 Prove que o produto cartesiano se distribui sobre a união e sobre a intersecção, ou seja, que (suponha A, B e C conjuntos quaisquer):

a) *Distributividade do produto cartesiano sobre a união.*

$$A \times (B \cup C) = (A \times B) \cup (A \times C)$$

b) *Distributividade do produto cartesiano sobre a intersecção.*

$$A \times (B \cap C) = (A \times B) \cap (A \times C)$$

Exercício 3.17 Por que a reversibilidade do produto cartesiano nem sempre é válida quando o conjunto resultante é vazio.

Dica: observe o EXEMPLO 3.9 - Produto Cartesiano.

Exercício 3.18 Prove que (suponha A, B e C conjuntos quaisquer):

a) $A \cup B = \varnothing \Rightarrow A = \varnothing \wedge B = \varnothing$

b) $A \cup B = B \wedge A \cap B = \varnothing \Rightarrow A = \varnothing$

c) $A \cap B = A \wedge B \cup C = C \Rightarrow A \cap \sim C = \varnothing$

d) $A \cap \sim B = \varnothing \wedge A \cap B = \varnothing \Rightarrow A = \varnothing$

e) $A = \varnothing \Leftrightarrow (\forall B)((A \cap \sim B) \cup (\sim A \cap B) = B)$

f) $A \subseteq B \Leftrightarrow \sim B \subseteq \sim A$

Exercício 3.19 Foi proposto, no Capítulo 2 - Lógica e Técnicas de Demonstração, um exercício o qual estabelece que qualquer dos conetivos estudados ($\neg$, $\wedge$, $\vee$, $\rightarrow$ e $\leftrightarrow$) pode ser expresso usando somente os conetivos $\neg$ e $\wedge$. Conseqüentemente, o mesmo vale para a Álgebra de Conjuntos, usando somente as operações $\sim$ e $\cap$, respectivamente. Então, qual a correspondência dos conetivos $\rightarrow$ e $\leftrightarrow$ na Álgebra de Conjuntos?

Exercício 3.20 Considere o EXEMPLO 3.14 - Programa em Pascal. Modifique o programa apresentado determinando, adicionalmente, o número de dígitos.

Exercício 3.21 Justifique os seguintes resultados:

a) A união de duas linguagens recursivas é uma linguagem recursiva;

Dica: considere os resultados já provados para as linguagens recursivas e lembre-se da propriedade de DeMorgan;

b) O complemento de uma linguagem recursivamente enumerável não necessariamente é uma linguagem recursivamente enumerável;

c) Um linguagem é recursiva se e somente se a linguagem e seu complemento são linguagens recursivamente enumeráveis.

Dica: o uso da propriedade de DeMorgan pode facilitar o desenvolvimento da solução.

Exercício 3.22 No Teorema 3.17 - Intersecção de Duas Linguagens Recursivas é Recursiva, por que se pode afirmar que a máquina $MT(L_1 \cap L_2)$ aceita a linguagem $L_1 \cap L_2$?

Dica: lembre-se da definição de intersecção e da relação entre a Lógica e a Álgebra de Conjuntos.

4 Relações

O conceito intuitivo de *relação* é muito próximo do conceito formal de relação. Os seguintes exemplos do cotidiano são relações:

- parentesco;
- "maior ou igual" (como estatura de pessoas);
- "igualdade" (como de números reais);
- lista telefônica que associa a cada assinante o seu número (ou números) de telefone;
- "faz fronteira com" para um conjunto de países;
- filas de pessoas para os diversos caixas em um banco.

Em Computação e Informática, muitas construções são baseadas em relações ou conceitos derivados (como *funções*), algumas das quais são introduzidas ao longo do livro. Entretanto, existem algumas importantes construções que podem ser definidas usando exclusivamente o conceito de relação como as seguintes, as quais são brevemente introduzidas neste capítulo:

- *Banco de Dados Relacional*;
- *Rede de Petri*.

Diversos conceitos relacionados com o de relação são desenvolvidos neste capítulo, como, por exemplo:

- *relação dual*, correspondendo à noção de "virar" ou "inverter" a relação;
- *composição* de relações, ou seja, a aplicação de uma relação sobre o resultado de outra;
- tipos de relações: *funcional*, *injetora*, *total*, *sobrejetora*, *monomorfismo*, *epimorfismo* e *isomorfismo*.

Eventualmente o leitor pode estar familiarizado com alguns dos tipos de relações citados acima. Entretanto, a abordagem que segue é mais voltada para Computação e Informática, diferindo um pouco daquela usualmente adotada em um contexto mais matemático. Exemplificando:

- na matemática tradicional, conceitos como injetora e sobrejetora são usualmente definidos para *funções*. Entretanto, como será visto adiante, na Computação e Informática, conceitos como *relação* e *função parcial* são tão ou mais importantes do que o de função. Portanto, tais conceitos devem ser tratados de forma mais geral;
- da mesma forma, na matemática tradicional, conceitos como funcional, injetora, total e sobrejetora são tratados de forma independente. Já na Computação e Informática, é desejável correlacioná-los usando uma noção de dualidade: funcional e total são os conceitos duais de injetora e sobrejetora, respectivamente. Como será visto adiante, tal correlação, entre outras vantagens, "divide o trabalho pela metade", no que se refere a definições, provas etc.

4.1 Relação

Além de freqüentemente usado no cotidiano, o conceito intuitivo de relação também é usual na Matemática e, conseqüentemente, na Computação e na Informática. No que já foi tratado, são exemplos de relações:

a) Teoria dos Conjuntos
- igualdade;
- continência;

b) Lógica
- equivalência;
- implicação.

Essas relações são ditas binárias, pois relacionam dois elementos de cada vez. Seguindo o mesmo raciocínio, existem relações ternárias, quaternárias, unárias, etc. Algumas relações podem ser definidas sobre coleções que não são conjuntos como, por exemplo, a continência sobre todos os conjuntos (que não é um conjunto, como já foi provado). O estudo que segue é centrado nas relações *binárias* e *pequenas*.

Definição 4.1 - Relação
Suponha A e B conjuntos. Uma *Relação* (pequena e binária) R de A em B é um subconjunto de um produto cartesiano $A \times B$, ou seja:

$$R \subseteq A \times B$$

sendo que:

A é denominado *domínio*, *origem* ou *conjunto de partida* de R
B é denominado *contradomínio*, *codomínio*, *destino* ou *conjunto de chegada* de R ❑

É importante observar que uma relação $R \subseteq A \times B$ é constituída de três partes: a origem A, o destino B e o conjunto de pares R. Qualquer alteração em uma destas três partes define uma outra relação.

Para uma relação $R \subseteq A \times B$, se $\langle a, b \rangle \in R$, afirma-se que:

$$a \text{ relaciona-se com } b$$

EXEMPLO 4.1 - Relação
Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então, são relações:

a) $\varnothing$ é uma relação de A em B, assim como de A em C, de B em C, etc. (exemplos diferentes de relações, com diferentes domínios e/ou codomínios), pois o conjunto vazio é subconjunto de qualquer conjunto;
b) $A \times B = \{\langle a, a \rangle, \langle a, b \rangle\}$ é uma relação com origem em A e destino B (pois $A \times B \subseteq A \times B$);
c) Considerando o conjunto de partida A e o conjunto de chegada B, a relação de igualdade é $\{\langle a, a \rangle\}$;
d) $\{\langle 0, 1 \rangle, \langle 0, 2 \rangle, \langle 1, 2 \rangle\}$ é a relação "menor que" de C em C;
e) $\{\langle 0, a \rangle, \langle 1, b \rangle\}$ é uma relação de C em B. ❑

Uma relação $R \subseteq A \times B$ também é denotada como segue:

$$R: A \rightarrow B$$

e um elemento $\langle a, b \rangle \in R$ é freqüentemente denotado de forma infixada, como segue:

$$a \, R \, b$$

EXEMPLO 4.2 - Relação: Notação
Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então:

a) Para a relação $\subseteq: \mathbf{P}(B) \rightarrow \mathbf{P}(B)$, tem-se que $\{a\} \subseteq \{a, b\}$
b) Para a relação $\leq: C \rightarrow C$, tem-se que $0 \leq 2$
c) Para a relação $=: A \rightarrow A$, tem-se que $a = a$ ❑

Observe que uma relação não necessariamente relaciona entidades de um mesmo conjunto. Por exemplo, uma lista telefônica relaciona pessoas (os assinantes) com números (de telefone). Entretanto, relacionar entidades de um mesmo conjunto é especialmente importante, razão pela qual recebe um nome próprio: *endorrelação*. Adicionalmente, seu estudo é detalhado em capítulo específico.

Definição 4.2 - Endorrelação, Auto-Relação
Suponha A um conjunto. Então uma relação $R: A \rightarrow A$ (origem e destino no mesmo conjunto) é dita uma *Endorrelação* ou *Auto-Relação*. Nesse caso, afirma-se que R é uma *relação em* A. ❑

Uma endorrelação $R: A \rightarrow A$ é freqüentemente denotada por:

$$\langle A, R \rangle$$

EXEMPLO 4.3 - Endorrelação
Seja A um conjunto. Então, as seguintes relações são endorrelações:

a) $\langle \mathbf{N}, \leq \rangle$
b) $\langle \mathbf{Z}, < \rangle$
c) $\langle \mathbf{Q}, = \rangle$
d) $\langle \mathbf{P}(A), \subseteq \rangle$
e) $\langle \mathbf{P}(\mathbf{R}), \subset \rangle$ ❑

Uma relação $R: A \rightarrow B$ pode ser representada usando *Diagrama de Venn*. Nesse caso, dois elementos relacionados são ligados por uma seta, com origem no elemento do conjunto de partida e destino no elemento de conjunto de chegada. Assim, na Figura 4.1, são ilustradas as seguintes situações (da esquerda para a direita):

- o par $\langle a, b \rangle$ da relação $R: A \rightarrow B$
- para $C = \{0, 1, 2\}$, a endorrelação $\langle C, < \rangle = \{\langle 0, 1 \rangle, \langle 0, 2 \rangle, \langle 1, 2 \rangle\}$. Observe que o conjunto C foi repetido no diagrama, sendo uma representação para a origem e outra para o destino. Tal fato é usual na diagramação de endorrelações.

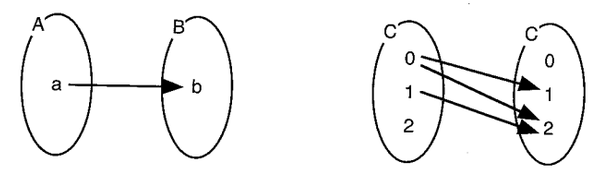


Figura 4.1 Relações representadas usando Diagrama de Venn

No estudo das relações, os termos definidos a seguir são importantes:

Definição 4.3 - Domínio de Definição, Conjunto Imagem
Seja $R: A \rightarrow B$ uma relação. Então:

a) Se $\langle a, b \rangle \in R$, então se afirma que R está *definida* para a, e que b é *imagem* de a;

b) O conjunto do todos os elementos de A para os quais R está definida é denominado de *Domínio de Definição*;

c) O conjunto do todos os elementos de B, imagem de R, é denominado de *Conjunto Imagem* ou *Domínio de Valores*. ❑

EXEMPLO 4.4 - Domínio de Definição, Conjunto Imagem

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então:

a) Para a relação $\varnothing: A \to B$, o domínio de definição e o conjunto imagem são vazios;

b) Para a endorrelação $\langle C, <\rangle$, dado que $<$ é definida por $\{\langle 0, 1\rangle, \langle 0, 2\rangle, \langle 1, 2\rangle\}$, o domínio de definição é $\{0, 1\}$, e o domínio de valores é $\{1, 2\}$;

c) Para a relação $=: A \to B$, o conjunto $\{a\}$ é o domínio de definição e o conjunto imagem. ❑

Uma representação especialmente interessante em diversas aplicações (computacionais ou não) é a representação gráfica das relações binárias e, em especial, das endorrelações em **R**. Nesse caso, os pares da relação $R \subseteq \mathbf{R}^2$ podem ser representados num plano, usualmente chamado de *plano cartesiano*. No exemplo que segue, é suposto que o leitor está familiarizado com esse tipo de representação.

EXEMPLO 4.5 - Representação de Relação no Plano Cartesiano

Considere as seguintes relações em **R**:

a) A relação trigonométrica seno, ilustrada na Figura 4.2 (esquerda):

$$R_1 = \{\langle x, y\rangle \mid y = \operatorname{sen} x\}$$

b) A figura geométrica parábola ilustrada na Figura 4.2 (direita):

$$R_2 = \{\langle x, y\rangle \mid x = y^2\}$$

Coordenadas polares são freqüentemente usadas para representar imagens no plano cartesiano. A partir de uma dada equação ou inequação, usando coordenadas polares, é possível retornar os pares do plano que definem a relação. Os casos que seguem são ilustrativos e não são discutidos:

c) A relação ilustrada na Figura 4.3 (esquerda) é definida a partir da seguinte inequação usando coordenadas polares:

$$\operatorname{sen}(6\cos r + 5\theta) < -0{,}3$$

d) A relação ilustrada na Figura 4.3 (direita) é definida a partir da seguinte inequação usando coordenadas polares:

$$\cos 10\theta < \tan(r \operatorname{sen} 2r)$$ ❑

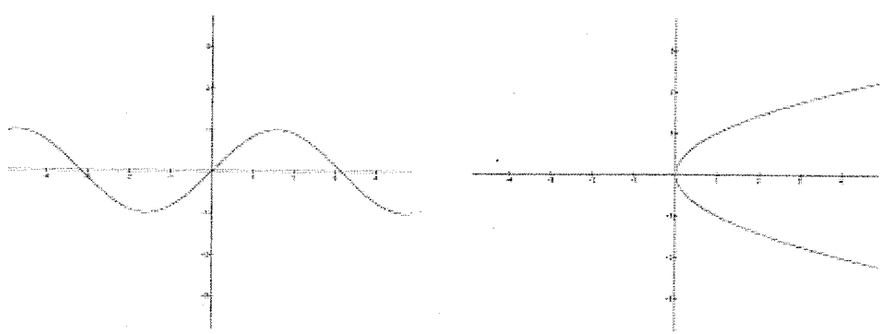


Figura 4.2 Relações representadas usando gráficos (equações)

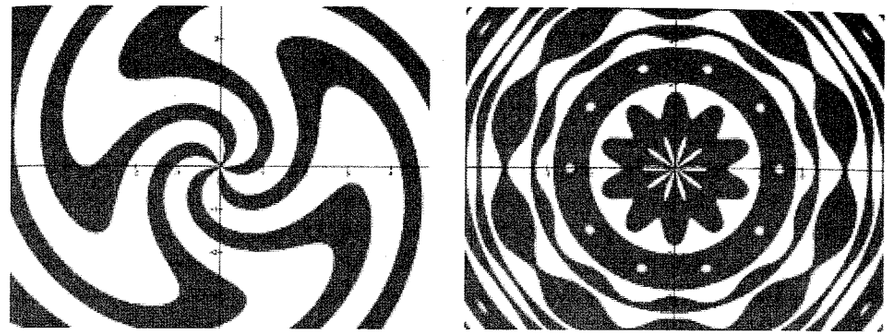


Figura 4.3 Relações representadas usando gráficos (inequações)

4.2 Endorrelação como Grafo

Toda endorrelação $R: A \to A$ pode ser representada como um *grafo*. De fato, como será visto adiante, toda endorrelação é um grafo, embora nem todo grafo seja uma endorrelação. Não é objetivo deste livro desenvolver uma Teoria dos Grafos, embora alguns conceitos sejam introduzidos ao longo de todo o livro (ver, em particular, o Capítulo 9 - Álgebras e Homomorfismos). Para a representação de relações que segue, algumas noções informais sobre grafos são suficientes.

Visualizar uma endorrelação como grafo muitas vezes facilita o seu estudo e fornece uma visão mais clara do relacionamento definido e de suas propriedades. Em geral, a representação física de uma relação como grafo (na forma de um desenho, por exemplo) é conveniente para relações com relativamente poucos pares (caso contrário, pode ficar confusa). Por outro lado, se a representação física não é importante, o tratamento de uma endorrelação como *grafo*, mesmo com um número infinito de pares, pode ser conveniente.

A representação de uma endorrelação $R: A \to A$ como *grafo* é como segue:

- cada elemento do conjunto A é representado como um ponto ou círculo, e é denominado de *nodo*;
- cada par $\langle a, b\rangle$ da relação é representado como uma *seta*, *arco* ou *aresta*, com origem em a e destino em b;

EXEMPLO 4.6 - Endorrelação como Grafo

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então as seguintes relações são representadas como grafos na Figura 4.4 (da esquerda para a direita):

a) $\varnothing: A \to A$ (grafo *sem* arestas)

b) $\langle B, =\rangle$, dado que $=$ é definida por $\{\langle a, a\rangle, \langle b, b\rangle\}$

c) $\langle C, <\rangle$, dado que $<$ é definida por $\{\langle 0, 1\rangle, \langle 0, 2\rangle, \langle 1, 2\rangle\}$

d) $R: C \to C$ tal que $R = \{\langle 0, 2\rangle, \langle 2, 0\rangle, \langle 2, 2\rangle\}$ ❑

É possível definir uma endorrelação para a qual o correspondente grafo possua dois ou mais arcos distintos com o mesmo nodo origem e destino? A resposta a esta questão ajuda a entender por que nem todo grafo é uma relação.

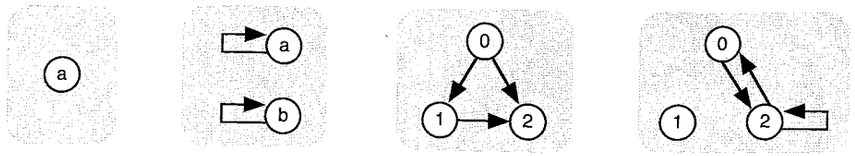


Figura 4.4 Endorrelações como grafos

4.3 Relação como Matriz

Suponha $A = \{a_1, a_2, \ldots, a_n\}$ e $B = \{b_1, b_2, \ldots, b_m\}$ dois conjuntos finitos. A representação de uma relação $R: A \to B$ na forma de matriz é especialmente interessante para implementação em um sistema computador. A representação é como segue:

- o número de linhas é n (número de elementos do domínio);
- o número de colunas é m (número de elementos do codomínio);
- portanto, a matriz resultante possui $m * n$ posições ou células;
- cada uma das $m * n$ posições da matriz contém um valor lógico (verdadeiro ou falso);
- se $\langle a_i, b_j \rangle \in R$, então a posição da matriz determinada pela linha i e coluna j contém o valor verdadeiro; caso contrário, contém o valor falso. Por simplicidade visual, nas matrizes que seguem, são usados os dígitos 0 e 1 para representar falso e verdadeiro, respectivamente.

EXEMPLO 4.7 - Relação como Matriz

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então as seguintes endorrelações são representadas como matrizes na Figura 4.5 (da esquerda para a direita):

a) $\varnothing: A \to A$ (como seria a relação $\varnothing: C \to C$?)
b) $\langle B, = \rangle$, dado que $=$ é definida por $\{\langle a, a\rangle, \langle b, b\rangle\}$
c) $\langle C, < \rangle$, dado que $<$ é definida por $\{\langle 0, 1\rangle, \langle 0, 2\rangle, \langle 1, 2\rangle\}$
d) $R: C \to C$ tal que $R = \{\langle 0, 2\rangle, \langle 2, 0\rangle, \langle 2, 2\rangle\}$

As seguintes relações são representadas como matrizes na Figura 4.6 (da esquerda para a direita):

e) $A \times B: A \to B$
f) $S = \{\langle 0, a\rangle, \langle 1, b\rangle\}: C \to B$
g) $\subseteq: \mathbf{P}(A) \to \mathbf{P}(B)$ ❑

$\varnothing$	a
a	0

=	a	b
a	1	0
b	0	1

<	0	1	2
0	0	1	1
1	0	0	1
2	0	0	0

R	0	1	2
0	0	0	1
1	0	0	0
2	1	0	1

Figura 4.5 Relações como matrizes

A×B	a	b
a	1	1

S	a	b
0	1	0
1	0	1
2	0	0

$\subseteq$	$\varnothing$	{a}	{b}	{a,b}
$\varnothing$	1	1	1	1
{a}	0	1	0	1

Figura 4.6 Relações como matrizes

4.4 Relação Dual e Composição de Relações

Duas questões são naturais no estudo das relações:

- a inversão (troca) das componentes de cada par ordenado que define uma relação, resultando em uma *relação dual*;
- a aplicação de uma relação sobre o resultado de outra, resultando em uma *relação composta*.

De fato, a dualidade e a composição de relações são operações (unária e binária, respectivamente) sobre as relações, as quais, juntamente com as operações sobre conjuntos como, por exemplo, união e intersecção (ver exercícios), constituem uma álgebra de relações. Trata-se de uma álgebra grande, pois a coleção de todas as relações não é um conjunto (a justificativa é sugerida como exercício).

4.4.1 Relação Dual

Definição 4.4 - Relação Dual, Relação Oposta, Relação Inversa

Seja $R: A \to B$ uma relação. Então a *Relação Dual*, *Relação Oposta* ou *Relação Inversa* de R, denotada por:

$$R^{-1}: B \to A \quad \text{ou} \quad R^{op}: B \to A$$

é como segue:

$$R^{-1} = \{\langle b, a\rangle \mid \langle a, b\rangle \in R\}$$ ❑

Observação 4.5 - Relação Dual × Relação Inversa

No texto que segue, o termo relação dual ou oposta será usado preferencialmente aos termos relação inversa. Embora referenciar relação inversa seja mais comum nas bibliografias em geral, o termo causa confusão com a idéia de uma relação que *possui inversa*, conceito fundamental no estudo das relações o qual será visto adiante. ❑

EXEMPLO 4.8 - Relação Dual

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então, em cada item abaixo, é apresentada uma relação (esquerda) e sua correspondente dual (direita):

$=: A \to B$ dada por $\{\langle a, a\rangle\}$	$=^{op}: B \to A$ dada por $\{\langle a, a\rangle\}$
$\{\langle 0, a\rangle, \langle 1, b\rangle\}: C \to B$	$\{\langle 0, a\rangle, \langle 1, b\rangle\}^{-1} = \{\langle a, 0\rangle, \langle b, 1\rangle\}: B \to C$
$A \times B: A \to B$	$(A \times B)^{-1} = B \times A: B \to A$
$\varnothing: A \to B$	$\varnothing^{op} = \varnothing: B \to A$
$<: C \to C$	$<^{op} = >: C \to C$

Como ilustração, a relação $<: C \to C$ e sua dual $<^{op}: C \to C$ são representadas na Figura 4.7 (esquerda e direita, respectivamente), usando Diagramas de Venn. ❑

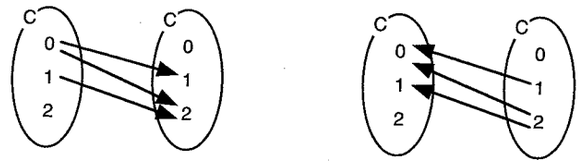


Figura 4.7 Relação e sua dual usando Diagramas de Venn

EXEMPLO 4.9 - Representação de Relação Dual no Plano Cartesiano
Compare o que segue com o EXEMPLO 4.5 - Representação de Relação no Plano Cartesiano. Então, em cada item abaixo, é apresentada uma relação em **R** e sua correspondente dual:

a) Relação trigonométrica seno

$R_1 = \{\langle x, y\rangle \mid y = \text{sen}\, x\}$
$R_1^{op} = \{\langle y, x\rangle \mid y = \text{sen}\, x\}$, representada na Figura 4.8 (esquerda);

b) Figura geométrica parábola

$R_2 = \{\langle x, y\rangle \mid x = y^2\}$
$R_2^{op} = \{\langle y, x\rangle \mid x = y^2\}$, representada na Figura 4.8 (direita). ❑

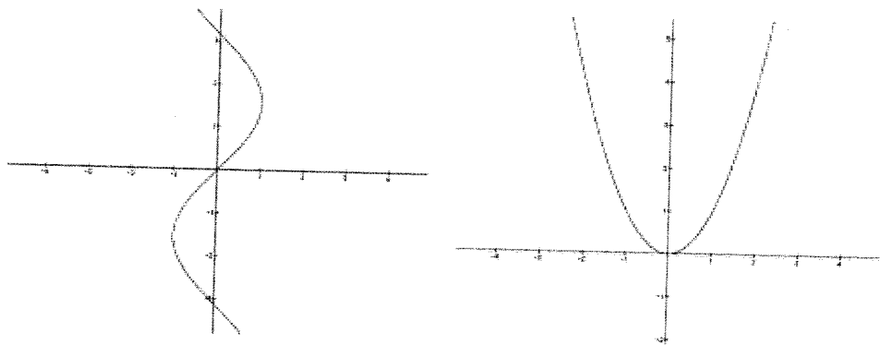


Figura 4.8 Relações duais representadas usando gráficos

Para uma dada relação, a correspondente relação dual como matriz ou como grafo (no caso de uma endorrelação) é como segue:

- matriz: a matriz da relação dual é a *matriz transposta* da matriz da relação, ou seja, é a matriz resultante da troca das linhas pelas colunas (e vice-versa);
- grafo: o grafo da endorrelação dual é o *grafo dual* da endorrelação, ou seja, é o grafo resultante da inversão do sentido das arestas.

EXEMPLO 4.10 - Grafo e Matriz de Relação Dual
Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$.

a) *Grafos*. Suponha as seguintes endorrelações, ilustradas na Figura 4.9 (esquerda):

$\langle C, <\rangle$
$\langle C, R\rangle$, sendo que R é definida por $\{\langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 1\rangle\}$

As correspondentes endorrelações duais são as seguintes, ilustradas na Figura 4.9 (direita):

$\langle C, <^{op}\rangle$, sendo que $<^{op}$ corresponde à relação $>$
$\langle C, R^{-1}\rangle$, sendo que R^{-1} é definida por $\{\langle 2, 1\rangle, \langle 1, 2\rangle, \langle 1, 0\rangle\}$

b) *Matrizes*. Suponha as seguintes relações, ilustradas na Figura 4.10 (esquerda):

$\langle C, <\rangle$
$\subseteq: \mathbf{P}(A) \to \mathbf{P}(B)$

Então, as correspondentes relações duais são as seguintes, as quais são ilustradas na Figura 4.10 (direita):

$\langle C, <^{op}\rangle$, sendo que $<^{op}$ corresponde à relação $>$ dada por $\{\langle 2, 1\rangle, \langle 2, 0\rangle, \langle 1, 0\rangle\}$
$\subseteq^{op}: \mathbf{P}(A) \to \mathbf{P}(B)$, sendo que $\subseteq^{op}$ corresponde à relação $\supseteq$ ❑

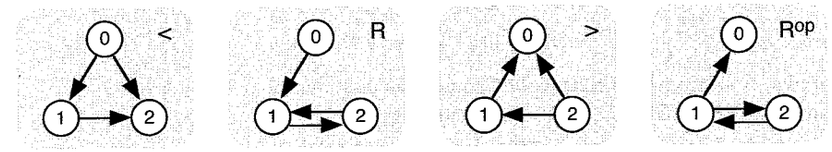


Figura 4.9 Grafos: endorrelações (esquerda) e as correspondentes endorrelações duais

<	0	1	2
0	0	1	1
1	0	0	1
2	0	0	0

⊆	∅	{a}	{b}	{a,b}
∅	1	1	1	1
{a}	0	1	0	1

>	0	1	2
0	0	0	0
1	1	0	0
2	1	1	0

⊇	∅	{a}
∅	1	0
{a}	1	1
{b}	1	0
{a,b}	1	1

Figura 4.10 Matrizes: relações (esquerda) e as correspondentes relações duais (direita)

4.4.2 Composição de Relações

Definição 4.6 - Composição de Relações
Sejam $R: A \to B$ e $S: B \to C$ relações. A *Composição* de R e S, resultando na relação composta denotada por:

$$S \circ R: A \to C$$

é tal que:

$$S \circ R = \{\langle a, c\rangle \mid (\exists b \in B)(a\, R\, b \wedge b\, S\, c)\}$$ ❑

Em Computação e Informática, é usual representar a operação de composição por ";". Nesse caso, para $R: A \to B$ e $S: B \to C$, tem-se que $S \circ R: A \to C$ é denotada em ordem inversa, isto é:

$$R\,;S: A \to C$$

EXEMPLO 4.11 - Composição de Relações
Considere a Figura 4.11. A composição das relações $R: A \to B$ e $S: B \to C$ é $S \circ R: A \to C$ sendo que:

$R = \{\langle a, 1\rangle, \langle b, 3\rangle, \langle b, 4\rangle, \langle d, 5\rangle\}$
$S = \{\langle 1, x\rangle, \langle 2, y\rangle, \langle 5, y\rangle, \langle 5, z\rangle\}$
$S \circ R = \{\langle a, x\rangle, \langle d, y\rangle, \langle d, z\rangle\}$ ❑

Obviamente, a relação resultante de uma composição pode ser composta com outra relação. Portanto, é importante investigar se a composição de relações é associativa.

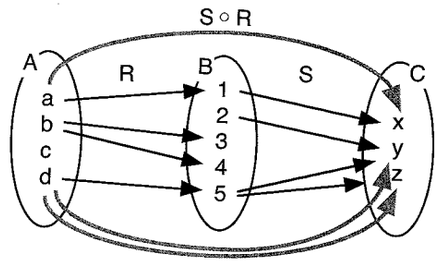


Figura 4.11 Composição de relações

Teorema 4.7 - Associatividade da Composição de Relações
Sejam $R: A \to B$, $S: B \to C$ e $T: C \to D$ relações. Então:

$$(T \circ S) \circ R = T \circ (S \circ R)$$

Prova: (*direta*)
Suponha que $R: A \to B$, $S: B \to C$ e $T: C \to D$ são relações. A prova é dividida em dois casos (duas continências), como segue:

Caso 1. $(T \circ S) \circ R \subseteq T \circ (S \circ R)$. Seja $\langle a, d\rangle \in (T \circ S) \circ R$

$\langle a, d\rangle \in (T \circ S) \circ R \Rightarrow$	definição de composição
$(\exists b \in B)(a\,R\,b \wedge b\,(T \circ S)\,d) \Rightarrow$	definição de composição
$(\exists b \in B)(\exists c \in C)(a\,R\,b \wedge (b\,S\,c \wedge c\,T\,d)) \Rightarrow$	associatividade da lógica
$(\exists b \in B)(\exists c \in C)((a\,R\,b \wedge b\,S\,c) \wedge c\,T\,d) \Rightarrow$	associatividade da lógica
$(\exists c \in C)(a\,(S \circ R)\,c \wedge c\,T\,d) \Rightarrow$	definição de composição
$\langle a, d\rangle \in T \circ (S \circ R)$	

Caso 2. $T \circ (S \circ R) \subseteq (T \circ S) \circ R$. A prova é análoga ao *caso 1* e é sugerida como exercício.
Logo, a composição de relações é associativa ❑

Portanto, como a composição de relações é associativa, os parênteses podem ser omitidos, ou seja:

$$(T \circ S) \circ R = T \circ (S \circ R) = T \circ S \circ R$$

O entendimento e a visualização da composição de relações como grafos não é especialmente vantajoso. Mas, como matrizes, a composição pode ser interpretada como o produto de matrizes. Entretanto, como os valores nas células das matrizes são lógicos (verdadeiro e falso, representados por 1 e 0, respectivamente), no cálculo do produto de matrizes, a multiplicação e a adição são substituídas pelos conetivos lógicos $\wedge$ e $\vee$, respectivamente. Assim, para as relações R e S representadas por matrizes $m \times n$ e $n \times p$, respectivamente, cada célula t_{uv} da matriz $m \times p$ resultante da composição, denotada por $T = S \circ R$, é calculada como segue (suponha que as células das matrizes R e S são denotadas por r e s juntamente com os correspondentes índices), respectivamente:

$$t_{uv} = (r_{u1} \wedge s_{1v}) \vee (r_{u2} \wedge s_{2v}) \vee \ldots \vee (r_{um} \wedge s_{mv})$$

EXEMPLO 4.12 - Composição de Relações como Produto de Matrizes
Considere o EXEMPLO 4.11 - Composição de Relações e a correspondente Figura 4.11. A composição das relações $R: A \to B$ (matriz 4×5) e $S: B \to C$ (matriz 5×3) é $T = S \circ R: A \to C$ (matriz 4×3) é ilustrada na Figura 4.12. Seguem os cálculos detalhados das células t_{11} e t_{23} (destacadas na matriz resultante):

$$t_{11} = (r_{11} \wedge s_{11}) \vee (r_{12} \wedge s_{21}) \vee (r_{13} \wedge s_{31}) \vee (r_{14} \wedge s_{41}) \vee (r_{15} \wedge s_{51}) =$$
$$(1 \wedge 1) \vee (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) \vee (0 \wedge 0) =$$
$$1 \vee 0 \vee 0 \vee 0 \vee 0 = 1$$

$$t_{23} = (r_{21} \wedge s_{13}) \vee (r_{22} \wedge s_{23}) \vee (r_{23} \wedge s_{33}) \vee (r_{24} \wedge s_{43}) \vee (r_{25} \wedge s_{53}) =$$
$$(0 \wedge 0) \vee (0 \wedge 0) \vee (1 \wedge 0) \vee (1 \wedge 0) \vee (0 \wedge 1) =$$
$$0 \vee 0 \vee 0 \vee 0 \vee 0 = 0$$

❑

R	1	2	3	4	5
a	1	0	0	0	0
b	0	0	1	1	0
c	0	0	0	0	0
d	0	0	0	0	1

S	x	y	z
1	1	0	0
2	0	1	0
3	0	0	0
4	0	0	0
5	0	1	1

T	x	y	z
a	1	0	0
b	0	0	0
c	0	0	0
d	0	1	1

Figura 4.12 Composição de relações como produto de matrizes

4.5 Tipos de Relações

Uma relação pode ser classificada nos seguintes tipos os quais não são mutuamente exclusivos:

- *funcional*;
- *injetora*;
- *total*;
- *sobrejetora*;
- *monomorfismo*;
- *epimorfismo*;
- *isomorfismo*.

Os tipos acima possuem uma noção de dualidade a qual, se for corretamente entendida e aplicada, pode simplificar ("dividir pela metade") o estudo e o entendimento dos diversos tipos de uma relação. De fato, tal noção de dualidade é plenamente explorada no estudo de *Teoria das Categorias*, razão pela qual se afirma que:

Teoria das Categorias divide o trabalho pela metade

Neste livro, somente algumas noções categoriais são introduzidas. A dualidade dos tipos de relação é como segue:

- funcional é o dual de injetora e vice-versa;
- total é o dual de sobrejetora e vice-versa.

Considerando que:

- *monomorfismo* significa simultaneamente total e injetora;
- *epimorfismo* significa simultaneamente sobrejetora e funcional;

vale, como conseqüência, a seguinte dualidade:

- monomorfismo é o dual de epimorfismo e vice-versa.

Por fim, *isomorfismo* estabelece uma noção semântica de igualdade e, portanto:

- isomorfismo é o dual de si mesmo.

4.5.1 Funcional e Injetora

Uma relação funcional é especialmente importante, pois permite definir *função*, conceito desenvolvido em capítulo específico.

Definição 4.8 - Relação Funcional
Seja $R: A \to B$ uma relação. Então R é uma *Relação Funcional* se e somente se:

$$(\forall a \in A)(\forall b_1 \in B)(\forall b_2 \in B)(a\,R\,b_1 \wedge a\,R\,b_2 \to b_1 = b_2) \qquad \square$$

Portanto, em uma relação funcional $R: A \to B$, cada elemento de A está relacionado com, no máximo, um elemento de B. A Figura 4.13 (esquerda) ilustra um contra-exemplo.

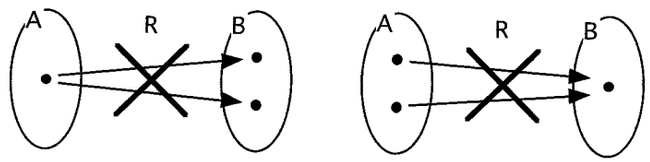


Figura 4.13 Contra-exemplo de relação funcional (esquerda) e injetora (direita)

EXEMPLO 4.13 - Relação Funcional
Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então (procure justificar cada um dos itens):

a) São relações funcionais:

$\varnothing: A \to B$
$\{\langle 0, a\rangle, \langle 1, b\rangle\}: C \to B$
$=: A \to B$
$x^2: \mathbf{Z} \to \mathbf{Z}$ onde $x^2 = \{\langle x, y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$

b) Não são relações funcionais:

$A \times B: A \to B$
$<: C \to C$ $\square$

Analisando uma relação funcional em termos da notação como matriz ou como grafo (no caso de uma endorrelação), tem-se que (como ficam os casos abaixo se a origem ou o destino for o conjunto vazio?):

- matriz: existe no máximo um valor verdadeiro em cada *linha* da matriz;
- grafo: existe no máximo uma aresta *partindo* de cada nodo.

Assim, considerando que *injetora* é o conceito dual, o seguinte pode ser inferido em termos da notação como matriz ou como grafo (no caso de uma endorrelação):

- matriz: existe no máximo um valor verdadeiro em cada *coluna* da matriz;
- grafo: existe no máximo uma aresta *chegando* a cada nodo.

Definição 4.9 - Relação Injetora
Seja $R: A \to B$ uma relação. Então R é uma *Relação Injetora* se e somente se:

$$(\forall b \in B)(\forall a_1 \in A)(\forall a_2 \in A)(a_1\,R\,b \wedge a_2\,R\,b \to a_1 = a_2) \qquad \square$$

Portanto, em uma relação injetora $R: A \to B$, cada elemento de B está relacionado com, no máximo, um elemento de A. A Figura 4.13 (direita) ilustra um contra-exemplo.

EXEMPLO 4.14 - Relação Injetora
Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então (procure justificar cada um dos itens):

a) São relações injetoras:

$\varnothing: A \to B$
$=: A \to B$
$\{\langle 0, a\rangle, \langle 1, b\rangle\}: C \to B$
$A \times B: A \to B$

b) Não são relações injetoras:

$B \times A: B \to A$
$<: C \to C$
$x^2: \mathbf{Z} \to \mathbf{Z}$ onde $x^2 = \{\langle x, y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$ $\square$

Observe o EXEMPLO 4.13 - Relação Funcional e o EXEMPLO 4.14 - Relação Injetora. Portanto, claramente, os conceitos funcional e injetora *não* são complementares. De fato, como ilustrado, é fácil encontrar exemplos de relações que são (respectivamente, não são) simultaneamente funcional e injetora.

4.5.2 Total e Sobrejetora

Definição 4.10 - Relação Total
Seja $R: A \to B$ uma relação. Então R é uma *Relação Total* se e somente se:

$$(\forall a \in A)(\exists b \in B)(a\,R\,b) \qquad \square$$

Portanto, em uma relação total $R: A \to B$, o domínio de definição é a origem A.

EXEMPLO 4.15 - Relação Total
Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então (procure justificar cada um dos itens):

a) São relações totais:

$=: A \to B$
$A \times B: A \to B$
$x^2: \mathbf{Z} \to \mathbf{Z}$ onde $x^2 = \{\langle x, y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$

b) Não são relações totais:

$\varnothing: A \to B$
$\{\langle 0, a\rangle, \langle 1, b\rangle\}: C \to B$
$<: C \to C$ $\square$

Analisando uma relação total em termos da notação como matriz ou como grafo (no caso de uma endorrelação), tem-se que (como ficam os casos abaixo se a origem ou o destino for o conjunto vazio?):

- matriz: existe pelo menos um valor verdadeiro em cada *linha* da matriz;
- grafo: existe pelo menos uma aresta *partindo* de cada nodo.

Assim, considerando que *sobrejetora* é o conceito dual, o seguinte pode ser inferido em termos da notação como matriz ou como grafo (no caso de uma endorrelação):

- matriz: existe pelo menos um valor verdadeiro em cada *coluna* da matriz;
- grafo: existe pelo menos uma aresta *chegando* a cada nodo.

Definição 4.11 - Relação Sobrejetora
Seja $R: A \to B$ uma relação. Então R é uma *Relação Sobrejetora* se e somente se:

$$(\forall b \in B)(\exists a \in A)(a\,R\,b) \qquad \square$$

Portanto, em uma relação sobrejetora $R: A \to B$, o conjunto imagem é B.

EXEMPLO 4.16 - Relação Sobrejetora

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então (procure justificar cada um dos itens):

a) São relações sobrejetoras:

$=: A \to A$
$\{\langle 0, a\rangle, \langle 1, b\rangle\}: C \to B$
$A \times B: A \to B$

b) Não são relações sobrejetoras:

$=: A \to B$
$\varnothing: A \to B$
$<: C \to C$
$x^2: \mathbf{Z} \to \mathbf{Z}$ onde $x^2 = \{\langle x, y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$ ❑

Observe o EXEMPLO 4.15 - Relação Total e o EXEMPLO 4.16 - Relação Sobrejetora. Portanto, claramente, os conceitos total e sobrejetora *não* são complementares. De fato, como ilustrado, é fácil encontrar exemplos de relações que são (respectivamente, não são) simultaneamente total e sobrejetora.

4.5.3 Monomorfismo e Epimorfismo

Monomorfismo, epimorfismo e isomorfismo são noções gerais que podem ser aplicadas a outras construções além das relações, como estudado em *Teoria das Categorias*. Entretanto, neste livro, esses conceitos serão restritos às relações.

Definição 4.12 - Monomorfismo, Monorrelação

Seja $R: A \to B$ uma relação. Então R é um *Monomorfismo* ou uma *Monorrelação* se e somente se for simultaneamente:

- total;
- injetora. ❑

Portanto, em uma monorrelação $R: A \to B$, o domínio de definição é A, e cada elemento de B está relacionado com, no máximo, um elemento de A. Para enfatizar que $R: A \to B$ é uma monorrelação, a seguinte notação é usual:

$$R: A \rightarrowtail B$$

EXEMPLO 4.17 - Monorrelação

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então (procure justificar cada um dos itens):

a) É monorrelação:

$=: A \to B$
$A \times B: A \to B$

b) Não são monorrelações:

$B \times C: B \to C$
$\varnothing: A \to B$
$\{\langle 0, a\rangle, \langle 1, b\rangle\}: C \to B$
$<: C \to C$
$x^2: \mathbf{Z} \to \mathbf{Z}$ onde $x^2 = \{\langle x, y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$ ❑

Analisando uma *monorrelação* em termos da notação como matriz ou como grafo (no caso de uma endorrelação), tem-se que:

- matriz: existe pelo menos um valor verdadeiro em cada *linha* (total) e no máximo um valor verdadeiro em cada *coluna* (injetora) da matriz;
- grafo: existe pelo menos uma aresta *partindo* (total) e no máximo uma aresta *chegando* (injetora) a cada nodo.

Assim, considerando que *epirrelação* é o conceito dual, ou seja, é uma relação funcional e sobrejetora, então o seguinte pode ser inferido em termos da notação como matriz ou como grafo (no caso de uma endorrelação):

- matriz: existe pelo menos um valor verdadeiro em cada *coluna* (sobrejetora) e no máximo um valor verdadeiro em cada *linha* (funcional) da matriz;
- grafo: existe pelo menos uma aresta *chegando* (sobrejetora) e no máximo uma aresta *partindo* (funcional) de cada nodo.

Definição 4.13 - Epimorfismo, Epirrelação

Seja $R: A \to B$ uma relação. Então R é um *Epimorfismo* ou uma *Epirrelação* se e somente se for simultaneamente:

- funcional;
- sobrejetora. ❑

Portanto, em uma epirrelação $R: A \to B$, o conjunto imagem é B e cada elemento de A está relacionado com, no máximo, um elemento de B. Para enfatizar que $R: A \to B$ é uma epirrelação, a seguinte notação é usual:

$$R: A \twoheadrightarrow B$$

EXEMPLO 4.18 - Epirrelação

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então (procure justificar cada um dos itens):

a) São epirrelações:

$=: A \to A$
$\{\langle 0, a\rangle, \langle 1, b\rangle\}: C \to B$

b) Não são epirrelações:

$=: A \to B$
$\varnothing: A \to B$
$A \times B: A \to B$
$<: C \to C$
$x^2: \mathbf{Z} \to \mathbf{Z}$ onde $x^2 = \{\langle x, y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$ ❑

4.5.4 Isomorfismo

Como já introduzido, o conceito de isomorfismo está associado a uma noção de igualdade semântica, no sentido em que se pode definir uma relação tal que, quando composta com a sua inversa, resulta em uma igualdade. Assim, intuitivamente, um isomorfismo possibilita "ir" (via relação) e "voltar" (via sua inversa), "sem alterar".

Para introduzir o conceito de isomorfismo, a seguinte notação é desejável:

- em diversos momentos, foram introduzidas relações de igualdade como, por exemplo:

$=:\{a\} \to \{a, b\}$ definida por $\{\langle a, a\rangle\}$
$\langle\{a, b\}, =\rangle$ definida por $\{\langle a, a\rangle, \langle b, b\rangle\}$

- quando a relação de igualdade é uma endorrelação $\langle A, =\rangle$, é usual denominar essa relação de *relação identidade* e denotá-la por $\langle A, id_A\rangle$ tal que:

$$id_A: A \to A$$

Definição 4.14 - Isomorfismo, Isorrelação

Uma relação $R: A \to B$ é dita um *Isomorfismo* ou uma *Isorrelação* se e somente se existe uma relação $S: B \to A$ tal que:

$$S \circ R = id_A \quad e \quad R \circ S = id_B$$ ❑

Para enfatizar que $R: A \to B$ é uma isorrelação, a seguinte notação é usual:

$$R: A \leftrightarrow B$$

Adicionalmente, vale que:

- quando $S \circ R = id_A$, afirma-se que R *possui inversa à esquerda*;
- quando $R \circ S = id_B$, afirma-se que R *possui inversa à direita*;
- quando $S \circ R = id_A$ e $R \circ S = id_B$, afirma-se que R (respectivamente S) *possui inversa*, a qual é S (respectivamente, R).

Definição 4.15 - Conjuntos Isomorfos

Dois conjuntos A e B são ditos *Conjuntos Isomorfos* se existe uma isorrelação $R: A \leftrightarrow B$. ❑

EXEMPLO 4.19 - Isomorfismo: Relação Identidade

Para qualquer conjunto A, a relação identidade $id_A: A \to A$ é um isomorfismo. De fato, a identidade composta com ela mesma resulta na própria identidade, ou seja:

$$id_A \circ id_A = id_A$$

Portanto, qualquer conjunto é isomorfo a si mesmo. ❑

EXEMPLO 4.20 - Isorrelação, Conjuntos Isomorfos

Sejam $A = \{a, b, c\}$, $C = \{1, 2, 3\}$ e a relação $R: A \to C$ tal que:

$$R = \{\langle a, 1\rangle, \langle b, 2\rangle, \langle c, 3\rangle\}$$

R é um isomorfismo. De fato, considere a relação dual $R^{-1}: C \to A$ tal que:

$$R^{-1} = \{\langle 1, a\rangle, \langle 2, b\rangle, \langle 3, c\rangle\}$$

Logo:

$$R^{-1} \circ R = \{\langle a, a\rangle, \langle b, b\rangle, \langle c, c\rangle\} = id_A \quad e \quad R \circ R^{-1} = \{\langle 1, 1\rangle, \langle 2, 2\rangle, \langle 3, 3\rangle\} = id_B$$

Portanto, A e C são conjuntos isomorfos. ❑

Assim, para provar que uma relação é um isomorfismo, basta mostrar que ela possui inversa. Observe que, *se* possui inversa, então esta é a relação dual. Entretanto, mostrar que uma relação *não* é um isomorfismo pode ser um pouco mais difícil (a dual nem sempre é a inversa). Freqüentemente, tal prova é por absurdo, como a ilustrada a seguir.

EXEMPLO 4.21 - Não é Isomorfismo

Sejam $A = \{0, 1, 2\}$, $B = \{a, b\}$ e a relação $R: A \to B$ tal que:

$$R = \{\langle 0, a\rangle, \langle 1, b\rangle\}$$

R *não* é um isomorfismo. A prova que segue é *por absurdo*. Assim, suponha que R *é* um isomorfismo. Portanto, R possui uma relação inversa $S: B \to A$ tal que:

$$S \circ R = id_A \quad e \quad R \circ S = id_B$$

Então:

$S \circ R = id_A \Rightarrow$ definição de relação identidade
$\langle 2, 2\rangle \in S \circ R \Rightarrow$ definição de composição
$(\exists x \in A)(\langle 2, x\rangle \in R \wedge \langle x, 2\rangle \in S)$

o que é um *absurdo*, pois não existe um par do tipo $\langle 2, x\rangle$ em R. Logo, é absurdo supor que R *é* um isomorfismo.

Portanto, R *não* é um isomorfismo. ❑

EXEMPLO 4.22 - Isorrelação

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então (procure provar cada um dos itens):

a) São isorrelações:

$\varnothing: \varnothing \to \varnothing$
$\{\langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 0\rangle\}: C \to C$
$ad_1: \mathbf{N} \to \mathbf{N} - \{0\}$ onde $ad_1 = \{\langle x, y\rangle \in \mathbf{N} \times \mathbf{N} - \{0\} \mid y = x + 1\}$
$R: \mathbf{Z} \to \mathbf{N}$ onde $R = \{\langle x, y\rangle \in \mathbf{Z} \times \mathbf{N} \mid y = 2x \text{ se } x \geq 0 \text{ e } y = |2x| - 1 \text{ se } x < 0\}$
sendo que $|2x|$ denota o módulo (valor absoluto) de $2x$

b) Não são isorrelações:

$\varnothing: A \to B$
$A \times B: A \to B$
$<: C \to C$
$x^2: \mathbf{Z} \to \mathbf{Z}$ onde $x^2 = \{\langle x, y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$ ❑

O seguinte teorema apresenta um importante resultado relativamente às isorrelações. A prova é omitida.

Teorema 4.16 - Isorrelação ↔ Monorrelação e Epirrelação

Seja $R: A \to B$ uma relação. Então R é uma isorrelação se e somente se, simultaneamente:

- R é uma monorrelação;
- R é uma epirrelação. ❑

Portanto, uma relação é uma isorrelação se e somente se for, simultaneamente:

- total;
- injetora;
- funcional;
- sobrejetora.

Esse resultado pode ser usado para auxiliar na determinação se uma determinada relação é ou não uma isorrelação. Como ilustração, compare o exemplo a seguir com o EXEMPLO 4.21 - Não é Isomorfismo.

EXEMPLO 4.23 - Não é Isomorfismo

Sejam $A = \{0, 1, 2\}$, $B = \{a, b\}$ e a relação $R: A \to B$ tal que:

$$R = \{\langle 0, a\rangle, \langle 1, b\rangle\}$$

R *não* é um isomorfismo. De fato, R não é total. ❑

Outro importante resultado que pode ser deduzido de um isomorfismo é que os conjuntos origem e destino possuem o mesmo número de elementos. Tal fato é explorado detalhadamente quando do estudo da *cardinalidade* (número de elementos) de um conjunto. Conjuntos que possuem o mesmo cardinal são ditos "iguais" (semanticamente) a menos de uma relação de troca

de nomes dos elementos, ou seja, a menos de uma isorrelação. Tal fato induz a seguinte afirmação, usual em muitas teorias, para um dado isomorfismo, para enfatizar que nomes de elementos não são importantes e que podem ser trocados:

iguais a menos de isomorfismo

EXEMPLO 4.24 - Igualdade a Menos de Isomorfismo: Produto Cartesiano

a) *Comutatividade.* Já foi visto que o produto cartesiano é uma operação não-comutativa. Por exemplo, para os conjuntos $A = \{a, b\}$ e $B = \{0, 1\}$, os seguintes conjuntos resultantes do produto cartesiano são claramente distintos:

$$A \times B = \{\langle a, 0\rangle, \langle a, 1\rangle, \langle b, 0\rangle, \langle b, 1\rangle\}$$
$$B \times A = \{\langle 0, a\rangle, \langle 0, b\rangle, \langle 1, a\rangle, \langle 1, b\rangle\}$$

Entretanto, são conjuntos isomorfos. De fato:

- considere a relação troca: $A \times B \to B \times A$ a qual troca a primeira componente de cada par pela segunda, ou seja:

$$\text{troca} = \{\langle\langle a, 0\rangle, \langle 0, a\rangle\rangle, \langle\langle a, 1\rangle, \langle 1, a\rangle\rangle, \langle\langle b, 0\rangle, \langle 0, b\rangle\rangle, \langle\langle b, 1\rangle, \langle 1, b\rangle\rangle\}$$

- a relação troca possui como inversa a relação dual $\text{troca}^{-1}: B \times A \to A \times B$ a qual simplesmente destroca as componentes, ou seja:

$$\text{troca}^{-1} = \{\langle\langle 0, a\rangle, \langle a, 0\rangle\rangle, \langle\langle 1, a\rangle, \langle a, 1\rangle\rangle, \langle\langle 0, b\rangle, \langle b, 0\rangle\rangle, \langle\langle 1, b\rangle, \langle b, 1\rangle\rangle\}$$

- claramente, o seguinte resultado ocorre:

$$\text{troca}^{-1} \circ \text{troca} = id_{A \times B} \quad \text{e} \quad \text{troca} \circ \text{troca}^{-1} = id_{B \times A}$$

A generalização deste raciocínio (sugerido como exercício) mostra que, para quaisquer dois conjuntos A e B, o conjunto $A \times B$ é isomorfo ao $B \times A$ e, portanto, esses conjuntos são *iguais a menos de isomorfismo*;

b) *Associatividade.* Raciocínio similar pode ser feito para a associatividade do produto cartesiano. Ou seja, o produto cartesiano é associativo, a menos de isomorfismo, ou seja, para quaisquer conjuntos A, B e C, vale que $(A \times B) \times C$ é isomorfo a $A \times (B \times C)$. A prova é sugerida como exercício. ❑

4.6 Banco de Dados Relacional

Bancos de dados são comuns na grande maioria das aplicações computacionais de algum porte ou de razoável complexidade (em termos dos dados), pois, além de permitir manipular os dados com maior eficiência e flexibilidade, atende a diversos usuários e garante a integridade (consistência) dos dados. Por integridade dos dados, entende-se que não existe a possibilidade de referenciar dados indevidos dentro da estrutura. Resumidamente, um banco de dados pode ser entendido como:

um conjunto de dados integrados cujo objetivo é atender a uma comunidade de usuários

Um *banco de dados relacional* é um banco de dados cujos dados são conjuntos (representados como tabelas) os quais são relacionados com outros conjuntos (tabelas).

EXEMPLO 4.25 - Banco de Dados Relacional

A Figura 4.14 representa um banco de dados relacional, sendo que:

- a tabela País denota o conjunto dos países considerados;
- a tabela Continente denota o conjunto dos continentes;
- a tabela Fica em denota uma relação com origem em País e destino em Continente. A corresponde matriz da relação é representada na Figura 4.15. Observe que um continente pode conter mais de um país, bem como um país pode estar em mais de um continente (como é o caso da Turquia).

Em um banco de dados relacional, podem ser definidas endorrelações como a ilustrada na Figura 4.16 (esquerda) a qual representa a relação semifinais e final da Copa do Mundo de Futebol de 2002, no conjunto País. A correspondente matriz da relação é representada na Figura 4.16 (direita). ❑

País
Brasil
Turquia
Alemanha
Coréia do Sul

Continente
América
Oceania
África
Ásia
Europa

Fica em	
Brasil	América
Alemanha	Europa
Turquia	Europa
Turquia	Ásia
Coréia do Sul	Ásia

Figura 4.14 Tabelas de um banco de dados relacional

Fica em	América	Oceania	África	Ásia	Europa
Brasil	1	0	0	0	0
Alemanha	0	0	0	0	1
Turquia	0	0	0	1	1
Coréia do Sul	0	0	0	1	0

Figura 4.15 Relação (como matriz) em um banco de dados relacional

Semifinais / Final Copa 2002	
Brasil	Turquia
Alemanha	Coréia do Sul
Brasil	Alemanha

Semifinais e Final Copa 2002	Bra	Tur	Ale	Cor
Brasil	0	1	1	0
Alemanha	1	0	0	1
Turquia	1	0	0	0
Coréia do Sul	0	0	1	0

Figura 4.16 Endorrelação em um banco de dados relacional: tabela (esquerda) e matriz (direita)

O projeto de um banco de dados é usualmente realizado usando um *modelo conceitual*, o qual é um modelo abstrato de dados que descreve a estrutura de um banco de dados independentemente de implementação. Um modelo conceitual freqüentemente adotado é o *diagrama entidade-relacionamento* ou simplesmente *diagrama E-R*. Assim, no exemplo em questão, tem-se que:

- *entidades*: conjuntos representados por retângulos, como ilustrado na Figura 4.17;

- *relacionamentos*: relacionamento entre as entidades, representados por losangos juntamente com traços que indicam (ligam) as entidades consideradas, como ilustrado na Figura 4.18.

Figura 4.17 Diagrama entidade-relacionamento: entidades

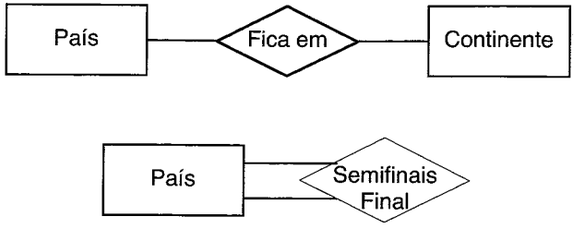


Figura 4.18 Diagrama entidade-relacionamento: relações

Em um diagrama E-R, é possível especificar o número de elementos que podem ser relacionados: 0, 1 ou N, denotando, zero, um ou mais que um, respectivamente. A especificação é realizada usando um par ordenado nos traços, indicando o número mínimo e máximo de elementos. Por exemplo, na Figura 4.19, tem-se que, para uma relação $R: A \rightarrow B$:

- (1, N) indica que cada elemento de A está relacionado com, no mínimo, 1 e, no máximo, N elementos de B;
- (0, 1) indica que cada elemento de B está relacionado com, no mínimo, 0 e, no máximo, 1 elemento de A;

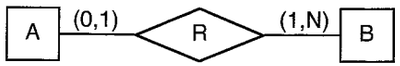


Figura 4.19 Especificação do número mínimo e máximo de elementos relacionados

Assim, na Figura 4.20, é ilustrado como alguns tipos de relações são representados em diagramas E-R (como seria o diagrama de uma monorrelação? E de uma epirrelação?).

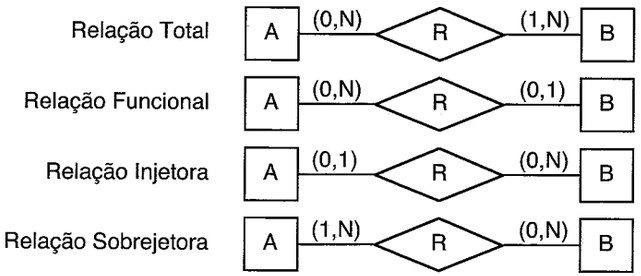


Figura 4.20 Tipos de relações em diagramas E-R

4.7 Rede de Petri

Redes de Petri constituem, provavelmente, o modelo computacional do tipo concorrente mais usado em Computação e Informática, bem como em diversas outras áreas, como, por exemplo, na engenharia em geral. O conceito de rede de Petri é introduzido através de exemplos. Após, é visto como qualquer rede de Petri pode ser definida como uma relação.

4.7.1 Modelo e Exemplos

EXEMPLO 4.26 - Rede de Petri - Produtor e Consumidor

Considere a rede de Petri ilustrada na Figura 4.21 a qual representa os processos concorrentes Produtor e Consumidor os quais trocam recursos através do Canal. Relativamente à rede:

- nodos (círculos) representam os *lugares* da rede;
- lugares podem conter *marcas* (em inglês, *tokens*), representadas por pequenos círculos pretos) os quais representam recursos disponíveis no sistema. Assim, na Figura 4.21, existe um recurso disponível para ser consumido no lugar A e um no lugar Y;
- arcos, representados por quadrados, são *transições* ou *computações atômicas*. Note-se que uma transição pode possuir mais de um lugar como origem ou destino. Adicionalmente, a cada lugar origem (destino) é associado um valor, representando quantos recursos serão consumidos (produzidos) quando da execução da transição. Por simplicidade, é usual omitir o valor quando este for 1;
- uma transição é dita uma *transição habilitada* se os recursos necessários para a execução estão disponíveis nos lugares origens;
- quando da execução de uma transição, os recursos especificados de cada lugar origem e destino são consumidos e produzidos, respectivamente. Assim, por exemplo, na execução da transição c_1 (se habilitada) são consumidas 3 marcas no lugar Canal e uma no lugar Y, e é gerada uma marca no lugar X. ❑

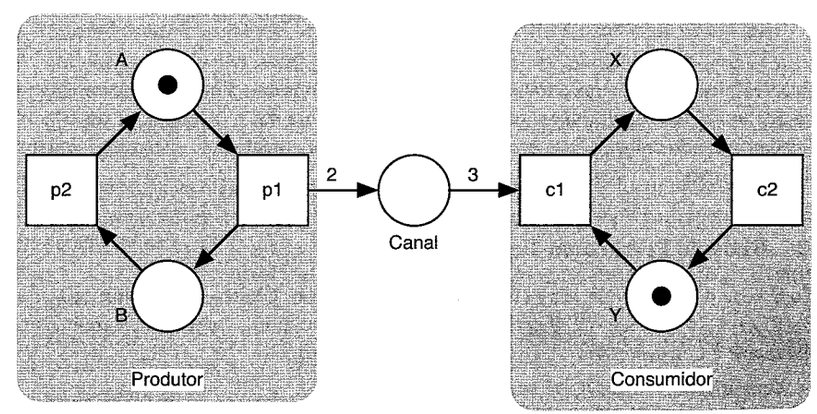


Figura 4.21 Rede de Petri - Produtor e Consumidor

O processamento de uma rede de Petri é a sucessiva aplicação de computações atômicas. Note-se que, em um mesmo instante, mais de uma transição pode estar habilitada. Nesse caso, elas podem ocorrer concorrentemente (desde que não consumam uma mesma marcação - neste caso, somente uma transição será executada). A Figura 4.22 ilustra uma seqüência de processamento, como segue:

- Figura 4.21: marcação inicial;
- Figura 4.22, quadro ①: p1 é executada, consumindo uma marca em A e gerando 2 marcas em Canal e uma marca em B;
- Figura 4.22, quadro ②: p2 é executada, consumindo uma marca em B e gerando uma marca em A;
- Figura 4.22, quadro ③: p1 é executada, consumindo uma marca em A e gerando 2 marcas em Canal e uma marca em B. Observe que Canal fica com 4 marcas, habilitando c1;
- Figura 4.22, quadro ④: p2 é executada, consumindo uma marca em B e gerando uma marca em A. Concorrentemente, c1 é executada, consumindo 3 marcas em Canal e uma marca em Y, gerando uma marca em X. Observe que ainda restou uma marca em Canal.

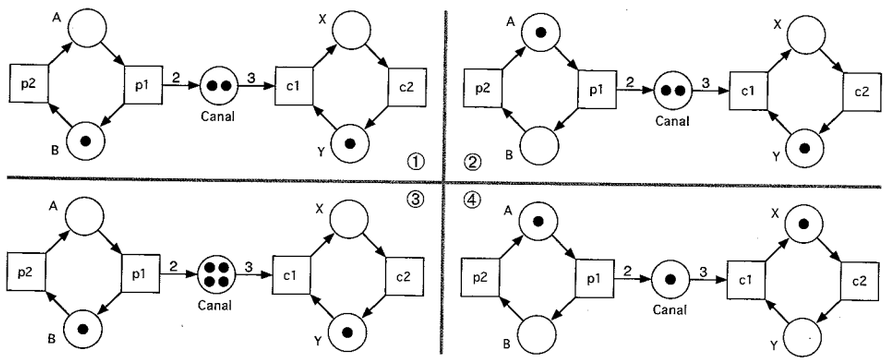


Figura 4.22 Seqüência de processamento de uma rede de Petri

Eventualmente duas ou mais transições podem tentar consumir simultaneamente uma mesma marca em um determinado lugar, como ilustrado na Figura 4.23. Neste caso, ou p1 ou p2 é executada (mas *não* ambas). Essa *escolha* (realizada aleatoriamente pelo sistema), é denominada *não-determinismo*.

4.7.2 Rede de Petri como Relação

Toda rede de Petri pode ser definida como uma relação. Observe como ilustração que, no EXEMPLO 4.26 - Rede de Petri - Produtor e Consumidor, ocorre a seguinte situação relativamente à transição p1:

- com origem no lugar A, p1 consome uma marcação;
- p1, com destino no lugar B, produz uma marcação;
- p1, com destino no lugar Canal, produz duas marcações;

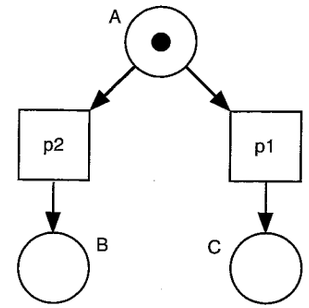


Figura 4.23 Rede de Petri - Não-determinismo

Portanto, na definição da transição p1, deve-se considerar os lugares origem e destino, bem como o número de marcações que são consumidas ou produzidas, respectivamente. Traduzindo os itens acima como pares ordenados, obtem-se, respectivamente:

$\langle\langle A, p1\rangle, 1\rangle$	p1 consome uma marcação no lugar A
$\langle\langle p1, B\rangle, 1\rangle$	p1 produz uma marcação no lugar B
$\langle\langle p1, Canal\rangle, 2\rangle$	p1 produz duas marcações no lugar Canal

Assim, cada par ordenado acima é tal que:

- a primeira componente é um par ordenado. Neste caso, se o lugar encontra-se antes da transição (respectivamente, depois) então se trata de um lugar origem (respectivamente, destino) da transição;
- a segunda componente é um valor natural indicando o número de marcações que são consumidas ou produzidas.

Portanto, supondo um conjunto de lugares L e um conjunto de transições T, uma rede de Petri pode ser definida como uma relação P como segue:

$$P: (L \times T) + (T \times L) \rightarrow \mathbf{N}$$

na qual (suponha que A é um lugar de L, t é uma transição de T e n é um número natural):

- um par da forma $\langle\langle A, t\rangle, n\rangle$ indica que a transição t, com *origem* em A, *consome* n marcações;
- um par da forma $\langle\langle t, A\rangle, n\rangle$ indica que a transição t, com *destino* em A, *produz* n marcações.

Observe que a união disjunta é usada para garantir que não há confusão entre os lugares e as transições (ou seja, é possível usar a mesma identificação para lugar ou para transição, que não ocorrerá ambigüidade).

Para o exemplo em questão, seguem todos os pares que definem a rede Produtor e Consumidor:

$\langle\langle A, p1\rangle, 1\rangle$ $\langle\langle p1, B\rangle, 1\rangle$ $\langle\langle p1, Canal\rangle, 2\rangle$	três pares para definir a transição p1
$\langle\langle B, p2\rangle, 1\rangle$ $\langle\langle p2, A\rangle, 1\rangle$	dois pares para definir a transição p2

⟨⟨Canal, c1⟩, 3⟩ três pares para definir a transição c1
⟨⟨Y, c1⟩, 1⟩
⟨⟨c1, X⟩, 1⟩

⟨⟨X, c2⟩, 1⟩ dois pares para definir a transição c2
⟨⟨c2, Y⟩, 1⟩

Observe que a relação define a rede, mas não a sua marcação inicial. De fato, uma mesma rede pode ser executada para diversas marcações iniciais. Portanto, a marcação inicial é, em geral, uma *opção* de execução e normalmente não faz parte da definição da rede.

EXEMPLO 4.27 - Rede de Petri - Relógio

Considere a rede de Petri ilustrada na Figura 4.21 a qual representa um relógio, considerando que:

- o relógio faz seqüências de tic-tac;
- cada tic-tac representa um segundo e move o ponteiro de segundos;
- a cada 60 segundos, move o ponteiro de minutos;
- a cada 3600 segundos, move o ponteiro das horas.

Observe que as transições que movem os ponteiros seg, min e horas consomem recursos dos lugares C, D e E, respectivamente, mas não produzem qualquer recurso em lugar algum. Transições desse tipo (que não produzem marcas) são chamadas de *sumidouros*. O caso dual (somente produzem marcas, sem consumir) são denominadas de *fontes*.

Seguem todos os pares que definem a rede Relógio:

⟨⟨A, tic⟩, 1⟩
⟨⟨tic, B⟩, 1⟩

⟨⟨B, tac⟩, 1⟩
⟨⟨tac, A⟩, 1⟩
⟨⟨tac, C⟩, 1⟩
⟨⟨tac, D⟩, 1⟩
⟨⟨tac, E⟩, 1⟩

⟨⟨C, seg⟩, 1⟩
⟨⟨D, min⟩, 60⟩
⟨⟨E, horas⟩, 3600⟩

Como seria se, a cada hora, um cuco desse uma badalada? E se o número de badaladas do cuco fosse o número de horas (entre 1 e 12 horas)? ❑

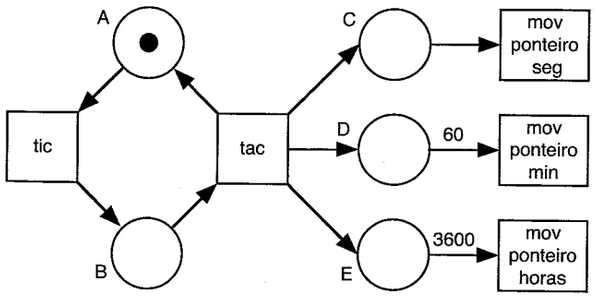


Figura 4.24 Rede de Petri - Relógio

4.8 Relações nas Linguagens de Programação

Relações não são normalmente disponíveis em linguagens de programação, excetuando-se em algumas construções simples as quais são predefinidas. No caso da linguagem Pascal, as seguintes construções já estudadas são exemplos de relações predefinidas:

- igualdade e continência de conjuntos;
- implicação e equivalência lógica;
- igualdade e desigualdade entre:

  valores numéricos (tipo `integer` e `real`);
  valores alfanuméricos (tipo `char`).

Entretanto, construções mais gerais usando relações como as estudadas neste capítulo necessitam ser construídas pelo próprio programador.

Assim, uma forma simples de implementar relações e construções correlatas (como a investigação de tipos, composição de relações, etc.), é usando relações como matrizes. Nesse caso, relações podem ser implementadas usando arranjos bidimensionais. Por exemplo, o seguinte trecho de programa em Pascal define uma matriz 10×10, sendo que cada célula da matriz é do tipo `boolean`:

```
matriz = array[1..10, 1..10] of boolean
```

Analogamente ao arranjo unidimensional já apresentado, cada componente pode ser diretamente acessada especificando-se o nome da variável arranjo seguido de dois índices (determinando a linha e coluna na matriz) entre colchetes. Por exemplo (suponha `i` e `j` variáveis do tipo `integer`):

```
matriz[2, 3] := true
```

```
if matriz[i, j] then ...
```

definem dois comandos como segue, respectivamente:

- atribui o valor `true` à componente posicionada na linha 2, coluna 3, da variável arranjo `matriz`;
- se a componente posicionada na linha `i`, coluna `j`, da variável arranjo `matriz` for verdadeira, então...

Uma importante restrição desse tipo de definição é que um arranjo possui um número fixo de componentes. Matrizes (e, conseqüentemente, relações) com número variável de células podem ser definidas usando outros tipos de construções as quais permitem alocação dinâmica de memória (espaço de armazenamento) como, por exemplo, ponteiros. Entretanto, construções com alocação dinâmica possuem particularidades dependentes de cada linguagem, e sua discussão foge do escopo desta publicação.

Uma implementação de relações como matrizes e construções correlatas é sugerida como exercício.

4.9 Exercícios

Exercício 4.1

a) Considere o EXEMPLO 4.5 - Representação de Relação no Plano Cartesiano. Para cada uma das seguintes relações, determine o domínio de definição bem como o conjunto imagem:

a.1) relação ilustrada na Figura 4.2 (esquerda);
a.2) relação ilustrada na Figura 4.2 (direita).

b) Analogamente para o EXEMPLO 4.9 - Representação de Relação Dual no Plano Cartesiano:

b.1) relação ilustrada na Figura 4.8 (esquerda);
b.2) relação ilustrada na Figura 4.8 (direita).

Exercício 4.2 Sejam $A = \{2, 3, 4, 5\}$ e $B = \{3, 4, 5, 6, 10\}$. Para cada uma das seguintes relações:

- explicite os elementos (pares) da relação;
- faça a representação gráfica (no plano cartesiano);
- determine o domínio de definição;
- determine o conjunto imagem.

a) $R_1 = \{\langle x, y\rangle \in A \times B \mid x \text{ é divisível por } y\}$
b) $R_2 = \{\langle x, y\rangle \in A \times B \mid x * y = 12\}$
c) $R_3 = \{\langle x, y\rangle \in A \times B \mid x = y + 1\}$
d) $R_4 = \{\langle x, y\rangle \in A \times B \mid x \leq y\}$

Exercício 4.3 As *relações* são fechadas para as seguintes operações sobre conjuntos (ou seja, a operação de duas relações resulta em uma relação)? Justifique a sua resposta:

a) União;
b) Intersecção;
c) Complemento;
d) Diferença;
e) Produto Cartesiano;
f) Conjunto das partes.

Exercício 4.4 É possível definir uma endorrelação para a qual o correspondente grafo possua dois ou mais *arcos paralelos* distintos, ou seja, os quais possuem os mesmos nodos origem e destino?

Exercício 4.5 A dualidade e a composição de relações são operações (unária e binária, respectivamente) sobre as relações, as quais, juntamente com as operações sobre conjuntos como, por exemplo, união e intersecção, constituem uma *álgebra de relações*. Trata-se de uma álgebra grande. Justifique por que a coleção de todas as relações não é um conjunto.

Dica: Para cada conjunto A, pode-se definir a endorrelação id_A, ou seja, para cada conjunto A existe pelo menos uma endorrelação.

Exercício 4.6 A prova do Teorema 4.7 - Associatividade da Composição de Relações é dividida em dois casos (duas continências). Detalhe a prova de que, de fato:

$$T \circ (S \circ R) \subseteq (T \circ S) \circ R$$

Exercício 4.7 Sejam $A = \{a\}$, $B = \{a, b\}$, $C = \{0, 1, 2\}$ e X um conjunto qualquer. Então, prove cada um dos seguintes itens:

a) São isorrelações:

a.1) $\varnothing : \varnothing \to \varnothing$
a.2) $\{\langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 0\rangle\} : C \to C$
a.3) $ad_1 : \mathbf{N} \to \mathbf{N} - \{0\}$ onde $ad_1 = \{\langle x, y\rangle \in \mathbf{N} \times \mathbf{N} - \{0\} \mid y = x + 1\}$
a.4) $R : \mathbf{Z} \to \mathbf{N}$ onde $R = \{\langle x, y\rangle \in \mathbf{Z} \times \mathbf{N} \mid y = 2x \text{ se } x \geq 0 \text{ e } y = |2x| - 1 \text{ se } x < 0\}$ sendo que $|2x|$ denota o módulo (valor absoluto) de $2x$

b) Não são isorrelações:

b.1) $\varnothing : A \to B$
b.2) $A \times B : A \to B$
b.3) $< : C \to C$
b.4) $x^2 : \mathbf{Z} \to \mathbf{Z}$ onde $x^2 = \{\langle x, y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$

Exercício 4.8 Generalize o EXEMPLO 4.24 - Igualdade a Menos de Isomorfismo: Produto Cartesiano, mostrando que, para quaisquer conjuntos A e B, tem-se que $A \times B$ é isomorfo a $B \times A$ e, portanto, são iguais a menos de isomorfismo.

Dica: Não esqueça de testar a sua solução para conjuntos vazios.

Exercício 4.9 Mostre que o produto cartesiano é associativo a menos de isomorfismo, ou seja, que, para quaisquer conjuntos A, B e C, vale que $(A \times B) \times C$ é isomorfo a $A \times (B \times C)$.

Exercício 4.10 "Elemento neutro a menos de isomorfismo":

a) Seja 1 um conjunto unitário. Mostre que, para qualquer conjunto A, o conjunto $A \times 1$ (ou $1 \times A$) é isomorfo ao próprio A e, portanto, são iguais a menos de isomorfismo. Assim, de certa forma, o conjunto unitário atua como um "elemento neutro" do produto cartesiano (a menos de isomorfismo);

b) Mostre que, para qualquer conjunto A, o conjunto $A + \varnothing$ (ou $\varnothing + A$) é isomorfo ao próprio A e, portanto, são iguais a menos de isomorfismo. Assim, de certa forma, o conjunto vazio atua como um "elemento neutro" da união disjunta (a menos de isomorfismo);

Exercício 4.11 Prove que, se uma relação possui inversa, então esta é única.

Exercício 4.12 Suponha uma relação $R : A \to B$. Como seria o diagrama E-R em cada uma das seguintes situações?

a) R é uma monorrelação;
b) R é uma epirrelação;
c) R é total e funcional (o que corresponde ao conceito de *função*, como será visto adiante);
d) R é total, funcional, injetora e sobrejetora (o que corresponde ao conceito de *função bijetora*, como será visto adiante).

Exercício 4.13 Considere a rede de Petri ilustrada na Figura 4.24, a qual representa um relógio. Modifique a rede considerando a existência de um cuco e prevendo as seguintes duas alternativas:

a) A cada hora, uma badalada do cuco;
b) A cada hora, o número de badaladas do cuco é o número de horas (entre 1 e 12 horas). ❑

Exercício 4.14 Uma ótima forma de entender corretamente, fixar e aplicar os conceitos estudados ao longo deste capítulo é implementá-los. O objetivo é fazer um pequeno sistema em que se possa definir relações e tratar construções correlatas como a investigação de seus tipos, cálculo da composição de relações, etc.

Assim, construa um pequeno sistema em qualquer linguagem de programação (Pascal, C, Java, ...) capaz de:

- definir até 6 conjuntos com até 5 elementos cada;
- definir até 5 relações como matrizes sobre estes conjuntos;
- verificar os tipos de uma relação;
- calcular a relação dual (e permitir verificar o tipo da relação dual);
- calcular (e armazenar) a composição de duas relações e permitir todas as ações acima sobre a relação resultante.

5 Funções Parciais e Totais

Uma *função parcial* é simplesmente uma relação funcional. Se a relação funcional for total, então é denominada de *função total* ou simplesmente *função*.

Portanto (veja a Figura 5.1):

- toda função é uma função parcial, bem como uma relação;
- toda função parcial é uma relação;
- nem toda relação é função parcial (de fato, basta considerar uma relação não-funcional);
- nem toda função parcial é uma função (de fato, basta considerar uma função parcial não-total).

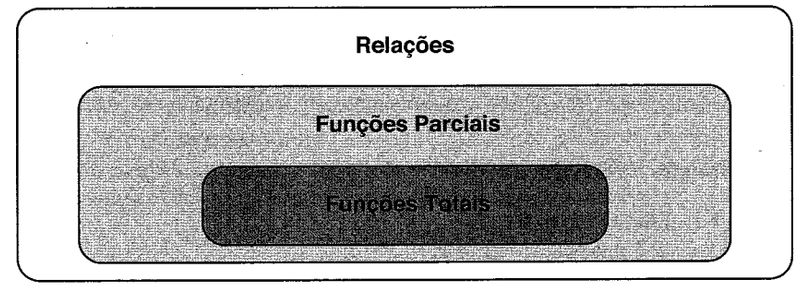


Figura 5.1 Continência própria

O estudo das funções é destacado do estudo das relações devido a sua importância para a Matemática em geral, e para a Computação e Informática em particular. Entretanto, ao contrário do que ocorre em muitas abordagens matemáticas centradas no conceito de função total, em Computação e Informática o conceito de função parcial é tão ou mais importante que o de função total. De fato, o conceito de *computabilidade*, que é claramente a noção mais fundamental nesse contexto, é baseado em funções parciais.

Neste capítulo, são discutidas diversas construções baseadas em funções parciais e totais e de interesse para Computação e Informática como, por exemplo:

- *operação de concatenação*;
- *multiconjunto*;
- *seqüência*;
- *conjunto indexado*.

Adicionalmente, as seguintes construções são especialmente destacadas:

- *Autômato Finito*;
- *Linguagem de Programação Funcional*.

5.1 Função Parcial

Já foi dito que uma função parcial é simplesmente uma relação funcional. Nesse contexto, todos os conceitos e terminologias introduzidos para relações que são válidos para relações funcionais são usuais para funções parciais como, por exemplo:

- as terminologias de domínio, imagem, etc.;
- os tipos injetora, sobrejetora, etc.

Alguns resultados são discutidos no contexto particular das funções parciais, com destaque para:

- condições para que a dual de uma função parcial seja uma função parcial;
- a prova de que a composição de funções parciais é uma função parcial.

De fato, a importância do correto entendimento da noção de dualidade, "dividindo o trabalho pela metade", vai aos poucos se revelando.

Por fim, é introduzida a operação de restrição (sobre funções parciais), a qual tem importantes aplicações em Ciência da Computação, como é ilustrado quando da introdução dos autômatos finitos.

Um exercício especialmente importante e interessante é o que trata de operações sobre funções (parciais ou totais) como produto cartesiano e união de funções.

5.1.1 Definição e Introdução

Definição 5.1 - Função Parcial

Uma *Função Parcial* é uma relação funcional $f \subseteq A \times B$. ❑

Portanto, uma função parcial é uma relação na qual cada elemento do domínio está relacionado com, no máximo, um elemento do contradomínio.

Uma função parcial $f \subseteq A \times B$ é denotada por:

$$f: A \nrightarrow B$$

ou, quando é claro que se trata de uma função parcial, simplesmente por:

$$f: A \rightarrow B$$

O par $\langle a, b\rangle \in f$ é usualmente denotado por:

$$f(a) = b \quad \text{ou} \quad f\langle a\rangle = b$$

Por simplicidade, quando a primeira componente do par $\langle a, b\rangle \in f$ é um elemento resultante de um produto cartesiano como, por exemplo, $a = \langle a_1, a_2\rangle$, tem-se que (observe a omissão dos parênteses):

$$f(\langle a_1, a_2\rangle) \text{ ou } f\langle\langle a_1, a_2\rangle\rangle \quad \text{é usualmente denotado por} \quad f(a_1, a_2) \text{ ou } f\langle a_1, a_2\rangle$$

Os seguintes termos também são usuais para denominar uma função parcial:

- *operação parcial*;
- *mapeamento parcial*;
- *transformação parcial*.

Considerando a definição de função parcial, os exemplos introduzidos como relações funcionais também são exemplos de funções parciais. Como reforço, alguns exemplos são reproduzidos a seguir.

EXEMPLO 5.1 - Função Parcial

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então (procure justificar cada um dos itens):

a) São funções parciais:

$\varnothing: A \rightarrow B$
$\{\langle 0, a\rangle, \langle 1, b\rangle\}: C \rightarrow B$
$=: A \rightarrow B$
$x^2: \mathbf{Z} \rightarrow \mathbf{Z}$ onde $x^2 = \{\langle x, y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$

b) Não são funções parciais:

$A \times B: A \rightarrow B$
$<: C \rightarrow C$ ❑

Lembre-se de que, em uma relação funcional (e, conseqüentemente, em uma função parcial), em termos da notação como matriz ou como grafo (no caso de uma endorrelação), tem-se que:

- matriz: existe no máximo um valor verdadeiro em cada *linha* da matriz;
- grafo: existe no máximo uma aresta *partindo* de cada nodo.

Seguem mais alguns exemplos de funções parciais.

EXEMPLO 5.2 - Função Parcial

a) *Adição nos naturais*. A operação $ad: \mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N}$ definida como abaixo é uma função parcial na qual o conjunto imagem é $\mathbf{N}$, ou seja, coincide com o contradomínio:

$$ad\langle a, b\rangle = a + b$$

b) *Divisão nos reais*. A operação $div: \mathbf{R} \times \mathbf{R} \rightarrow \mathbf{R}$ definida como abaixo é uma função parcial (qual o conjunto imagem?):

$$div\langle x, y\rangle = x/y$$ ❑

5.1.2 Função Parcial Dual

A relação dual de uma função parcial não necessariamente é uma função parcial, como ilustrado no seguinte exemplo.

EXEMPLO 5.3 - Relação Dual de Função Parcial

Sejam $A = \{0, 1, 2\}$ e a endofunção parcial $f: A \rightarrow A$ tal que $f = \{\langle 0, 2\rangle, \langle 1, 2\rangle\}$. Então:

$$f^{op} = \{\langle 2, 0\rangle, \langle 2, 1\rangle\}$$

é claramente uma relação não-funcional. ❑

Lembre-se de que o conceito dual de funcional é injetora. Assim, para que a dual de uma relação funcional (função parcial) seja uma função parcial, esta deve ser injetora. Portanto:

a relação dual de uma relação funcional e injetora,
é uma relação injetora e funcional.

Logo:

a relação dual de uma função parcial injetora,
é uma função parcial injetora.

EXEMPLO 5.4 -Relação Dual de Função Parcial
Sejam $A=\{a\}$, $B=\{a,b\}$ e $C=\{0,1,2\}$. Então:

a) As relações duais das seguintes funções parciais são funções parciais:

$\varnothing: A \to B$
$=: A \to B$
$\{\langle 0,a\rangle, \langle 1,b\rangle\}: C \to B$

b) A relação dual da seguinte função parcial *não* é uma função parcial (por quê?):

$x^2: \mathbf{Z} \to \mathbf{Z}$ onde $x^2 = \{\langle x,y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$ ❑

É ISSO QUE EU QUERO SABER!

5.1.3 Composição de Funções Parciais

Por definição, a composição de relações é uma relação. Entretanto, a composição de relações preserva a funcionalidade? Ou seja, a composição de funções parciais é uma função parcial? Para provar esse resultado, basta mostrar que a composição de funcionais é funcional.

Teorema 5.2 - Composição de Funcionais é Funcional
Sejam $R: A \to B$ e $S: B \to C$ relações funcionais. Então a relação composta $S \circ R: A \to C$ é uma relação funcional.

Prova:
Suponha que $R: A \to B$ e $S: B \to C$ são relações funcionais. Como a composição de relações é uma relação, então $S \circ R: A \to C$ é uma relação. Portanto, basta provar que a composição de relações funcionais é funcional, ou seja, que $S \circ R: A \to C$ é tal que:

$$(\forall a \in A)(\forall c_1 \in C)(\forall c_2 \in C)(a\,(S \circ R)\,c_1 \wedge a\,(S \circ R)\,c_2 \to c_1 = c_2)$$

De fato, suponha $a \in A$, $c_1 \in C$ e $c_2 \in C$ tais que $a\,(S \circ R)\,c_1 \wedge a\,(S \circ R)\,c_2$. Então (prova direta):

$a\,(S \circ R)\,c_1 \wedge a\,(S \circ R)\,c_2 \Rightarrow$	definição de composição
$(\exists b_1 \in B)(\exists b_2 \in B)(a\,R\,b_1 \wedge a\,R\,b_2 \wedge b_1\,S\,c_1 \wedge b_2\,S\,c_2) \Rightarrow$	R é funcional
$b_1 = b_2 \wedge b_1\,S\,c_1 \wedge b_2\,S\,c_2 \Rightarrow$	S é funcional
$c_1 = c_2$	

Logo, $S \circ R: A \to C$ é uma relação funcional. ❑

Portanto, a composição de funções parciais é uma função parcial.

EXEMPLO 5.5 - Composição de Funções Parciais
Considere a Figura 5.2. A composição das funções parciais $f: A \to B$ e $g: B \to C$ é $g \circ f: A \to C$ sendo que:

$f = \{\langle a,1\rangle, \langle c,5\rangle, \langle d,5\rangle\}$
$g = \{\langle 1,x\rangle, \langle 2,y\rangle, \langle 4,y\rangle, \langle 5,z\rangle\}$
$g \circ f = \{\langle a,x\rangle, \langle c,z\rangle, \langle d,z\rangle\}$ ❑

Observação 5.3 - Dualidade e Prova de Teoremas
Um fato extremamente importante é que todo resultado válido para um dado conceito também é válido para o seu conceito dual, e a prova é praticamente a mesma, respeitando as noções duais. Tal fato pode não ser tão evidente na Teoria dos Conjuntos, mas é amplamente explorado na *Teoria das Categorias*, na qual, como já foi dito:

a noção de dualidade divide o trabalho pela metade

incluindo definições e provas. ❑

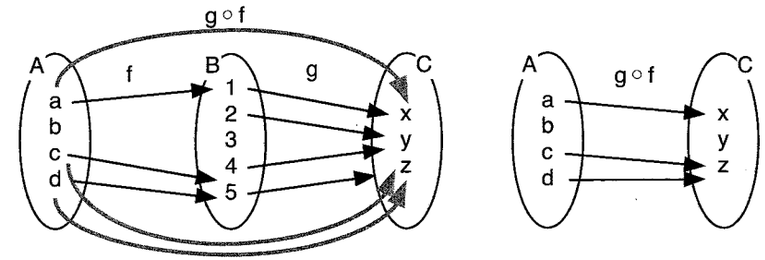


Figura 5.2 Composição de funções parciais

Considerando a observação acima, por dualidade, a composição de relações injetoras é uma relação injetora. Para ilustrar tal fato, sugere-se, como exercício, fazer a prova do seguinte corolário, "dualizando" a prova do Teorema 5.2 - Composição de Funcionais é Funcional.

Corolário 5.4 - Composição de Injetoras é Injetora
Sejam $R: A \to B$ e $S: B \to C$ relações injetoras. Então a relação composta $S \circ R: A \to C$ é uma relação injetora. ❑

5.1.4 Restrição

Para uma dada função parcial, é possível definir uma restrição da mesma, a partir de um subconjunto de seu domínio. Trata-se de uma operação (sobre funções) importante quando aplicada sobre sistemas, sendo uma das operações fundamentais do que é denominado de *álgebra de processos*.

Definição 5.5 - Restrição do Domínio de uma Função Parcial
Sejam $f: A \to B$ uma função parcial e A_0 um conjunto tal que $A_0 \subseteq A$. Então, a *Restrição do Domínio de* f relativamente a A_0, denotada por:

$$f\backslash A_0: A_0 \to B$$

é tal que:

$$f\backslash A_0 = f \cap (A_0 \times B)$$ ❑

Portanto, para uma dada $f: A \to B$ e para um dado subconjunto do domínio $A_0 \subseteq A$, a função resultante da restrição $f\backslash A_0$, como o próprio nome indica, é simplesmente a função f restringindo o seu domínio a A_0.

EXEMPLO 5.6 - Restrição do Domínio de uma Função Parcial
Sejam $A = \{1,2,3,4,5\}$, $B = \{x,y,z\}$ e a função parcial $f: A \to B$ como ilustrado na Figura 5.3 (esquerda). Para $A_0 = \{3,4,5\}$, $f\backslash A_0: A_0 \to B$ é como ilustrado na Figura 5.3 (direita).

EXEMPLO 5.7 - Restrição do Domínio de uma Função Parcial
Sejam $A = \{a\}$, $B = \{a,b\}$ e $C = \{0,1,2\}$. Então, para cada função parcial na coluna da esquerda é apresentada, na coluna da direita, uma correspondente função parcial restrita para um dado subconjunto do domínio (procure justificar cada um dos itens):

a) $\varnothing: A \to B$	$\varnothing\backslash A = \varnothing: A \to B$
b) $R = \{\langle 0,a\rangle, \langle 1,b\rangle\}: C \to B$	$R\backslash\{0\} = \{\langle 0,a\rangle\}: \{0\} \to B$
c) $id_B = \{\langle a,a\rangle, \langle b,b\rangle\}: B \to B$	$id_B\backslash A = \{\langle a,a\rangle\}: A \to B$
d) $x^2 = \{\langle x,y\rangle \in \mathbf{Z}^2 \mid y = x^2\}: \mathbf{Z} \to \mathbf{Z}$	$x^2\backslash\mathbf{N} = \{\langle x,y\rangle \in \mathbf{N} \times \mathbf{Z} \mid y = x^2\}: \mathbf{N} \to \mathbf{Z}$

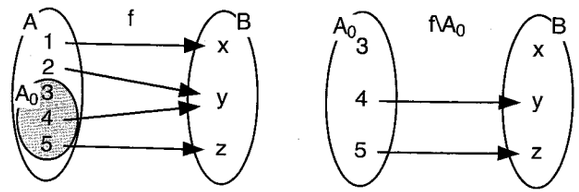


Figura 5.3 Função parcial (esquerda) e sua correspondente restrição (direita)

❑

A restrição introduzida foi sobre o domínio. Claramente, poderia ser definida sobre o contradomínio, o que é sugerido como exercício.

Uma exemplificação de como uma restrição de função parcial pode ser usada para calcular a restrição de sistemas é apresentada na próxima secção.

5.2 Autômato Finito

Autômato Finito é um sistema de estados finitos e pré-definidos o qual constitui um modelo computacional do tipo seqüencial muito comum em diversos estudos teórico-formais da Ciência da Computação, com destaque para *Linguagens Formais*, *Compiladores*, *Semântica Formal* e *Teoria da Concorrência*. O conceito de autômato finito será introduzido através de exemplos e é baseado na definição usualmente adotada em *Linguagens Formais* onde é usado para verificar se uma palavra w pertence a uma linguagem L, ou seja, o autômato verifica se:

$$w \in L \quad \text{ou} \quad w \notin L$$

5.2.1 Modelo e Exemplo

Um *Autômato Finito* pode ser visto como uma máquina composta, basicamente, de três partes:

a) *Fita*. Dispositivo de entrada que contém a informação a ser processada;
b) *Unidade de Controle*. Reflete o estado corrente da máquina. Possui uma unidade de leitura, denominada *cabeça da fita*, a qual acessa uma célula da fita de cada vez e movimenta-se exclusivamente para a direita;
c) *Programa* ou *Função de Transição*. Função que comanda as leituras e define o estado da máquina.

A fita é finita (à esquerda e à direita), sendo dividida em células, cada uma das quais armazena um símbolo. Os símbolos pertencem a um alfabeto de entrada. Não é possível gravar sobre a fita (e não existe memória auxiliar). Inicialmente, a palavra a ser processada (ou seja, a informação de entrada para a máquina) ocupa toda a fita.

A unidade de controle possui um número finito e predefinido de estados. A unidade de leitura lê o símbolo de uma célula de cada vez. Após a leitura, a cabeça da fita move-se uma célula para a direita. Inicialmente, a cabeça está posicionada na célula mais à esquerda da fita, como ilustrado na Figura 5.4.

ESCOLA DE ARTES, CIÊNCIAS E HUMANIDADES - USP
— BIBLIOTECA — 5213

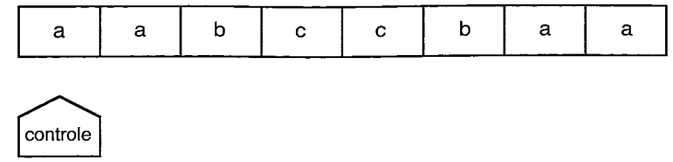


Figura 5.4 Autômato finito como uma máquina com controle finito

O programa é uma função parcial tal que:

dependendo do estado corrente e do símbolo lido,
determina o novo estado do autômato.

EXEMPLO 5.8 - Autômato Finito

Considere o autômato finito ilustrado na Figura 5.5 (esquerda) onde:

- nodos representam *estados* do autômato, os quais são em número *finito*;
- arcos representam *transições* ou *computações atômicas* as quais são em número *finito*. O conjunto de todas as transições constitui o programa do autômato;
- o estado q_0 é dito *estado inicial* e é representado de forma diferenciada (no caso, o nodo destino de uma seta sem origem);
- o estado q_f é dito *estado final* e é representado de forma diferenciada (no caso, o traço da circunferência do nodo é mais forte). ❑

O processamento de um autômato finito é a sucessiva aplicação de computações atômicas. No caso de Linguagens Formais, o autômato pára (normalmente) quando processar toda a informação de entrada. Nesse caso, aceita a entrada se parar em um estado final.

O autômato da Figura 5.5 (esquerda) aceita todas as palavras sobre o alfabeto $\{a, b\}$ que possuem aa ou bb como subpalavra. A Figura 5.5 (direita) ilustra o processamento do autômato em questão para a entrada w = abba, a qual é aceita.

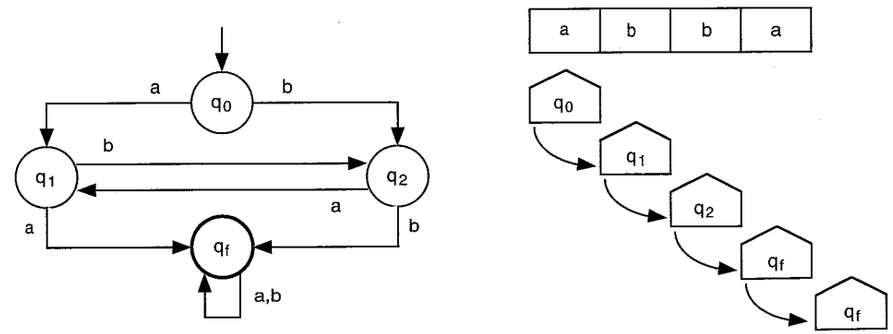


Figura 5.5 Autômato finito (esquerda) e correspondente seqüência de processamento (direita)

5.2.2 Autômato Finito como Função Parcial

Todo autômato finito pode ser definido como uma função parcial. Observe, como ilustração, que, no EXEMPLO 5.8 - Autômato Finito, ocorre a seguinte situação relativamente ao estado q_0:

- com origem no estado q_0, ao ler o símbolo a, o autômato assume o estado q_1;
- com origem no estado q_0, ao ler o símbolo b, o autômato assume o estado q_2.

Portanto, dependendo do estado corrente e do símbolo lido da fita, o autômato assume um novo estado. Traduzindo os itens acima como pares ordenados, obtem-se, respectivamente:

$\langle\langle q_0, a\rangle, q_1\rangle$
$\langle\langle q_0, b\rangle, q_2\rangle$

Assim, cada par ordenado acima é tal que:

- a primeira componente é um par ordenado, o qual define o estado corrente e o símbolo lido da fita;
- a segunda componente é o novo estado.

Portanto, supondo um conjunto finito de estados Q e um alfabeto Σ, o programa de um autômato finito pode ser definido como uma função parcial, denominada de *função programa*:

$$\delta: Q \times \Sigma \rightarrow Q$$

na qual um par da forma $\langle\langle q, a\rangle, p\rangle$, ou seja, tal que $\delta\langle q, a\rangle = p$, indica que, no estado q, ao ler o símbolo a, o autômato assume o estado p.

Para o autômato da Figura 5.5, seguem todos os pares que definem a função programa do autômato finito (a função parcial é total? É injetora? É sobrejetora?):

$\langle\langle q_0, a\rangle, q_1\rangle$	$\langle\langle q_1, a\rangle, q_f\rangle$	$\langle\langle q_2, a\rangle, q_1\rangle$	$\langle\langle q_f, a\rangle, q_f\rangle$
$\langle\langle q_0, b\rangle, q_2\rangle$	$\langle\langle q_1, b\rangle, q_2\rangle$	$\langle\langle q_2, b\rangle, q_f\rangle$	$\langle\langle q_f, b\rangle, q_f\rangle$

Observe que a função parcial define o programa do autômato, o qual não inclui o estado inicial nem os finais.

Uma aplicação usual dos autômatos finitos está na definição de *interfaces* de comunicação homem × máquina, como ilustrado no exemplo a seguir.

EXEMPLO 5.9 - Autômato Finito como Interface Homem × Máquina

Considere o autômato finito ilustrado na Figura 5.6 (esquerda) o qual representa a *interface* homem × máquina de uma máquina de vendas de refrigerante, cigarro e doce, definida pela seguinte função programa (a função parcial é total? É injetora? É sobrejetora?):

$$\delta: Q \times \Sigma \rightarrow Q$$

supondo que:

$Q = \{q_0, q_1, q_2, q_3, q_4\}$
$\Sigma = \{$moeda, tecla_doce, tecla_cigarro, tecla_refri, libera_doce, libera_cigarro, libera_refri$\}$

❑

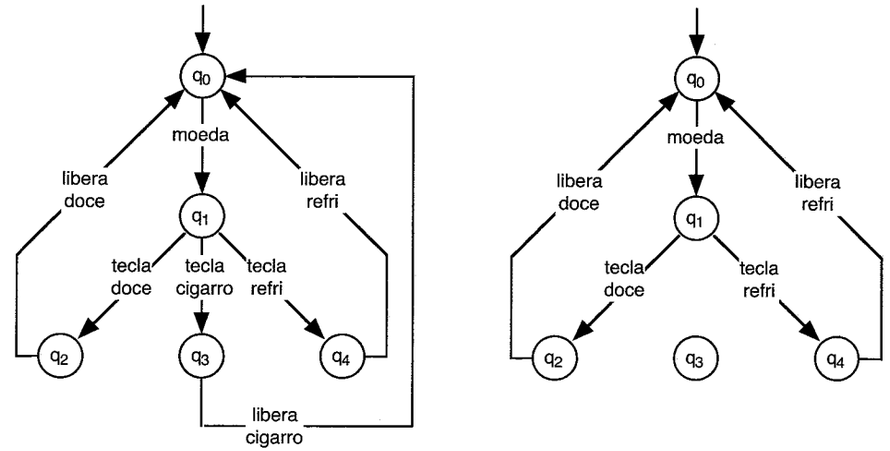


Figura 5.6 Autômato finito como interface homem × máquina e a correspondente restrição

5.2.3 Restrição de um Autômato Finito

Como destacado anteriormente, uma importante aplicação da operação de restrição de funções parciais é no cálculo de restrição de sistemas, determinando uma noção de *reuso de software*, importante conceito da *Engenharia de Software* e do paradigma *Orientação a Objetos*.

EXEMPLO 5.10 - Restrição de Autômato Finito × Reuso de Software

Considere o autômato finito ilustrado na Figura 5.6 (esquerda). Suponha que é desejada uma nova máquina, análoga a esta, mas sem as funções relacionadas com cigarros, como ilustrado na Figura 5.6 (direita). Tal máquina pode ser obtida, simplesmente realizando uma operação de restrição, como segue:

$$\delta \backslash Q \times \Sigma_0 : Q \times \Sigma_0 \rightarrow Q$$

supondo que:

$$\Sigma_0 = \{\text{moeda, tecla_doce, tecla_refri, libera_doce, libera_refri}\}$$

Sugere-se como exercício detalhar a função programa deste autômato. ❑

Observação 5.6 - Manutenção de Software

Uma operação de restrição de *software* pode facilmente ser implementada como uma ferramenta automática de desenvolvimento e de manutenção de sistemas. Ou seja, o programador *não* altera diretamente o *software*, mas sim realiza uma operação sobre este, garantindo automaticamente o resultado desejado. Destaque-se que o custo de manutenção de *software* é, com freqüência, maior que o de desenvolvimento (ao longo do ciclo de vida de um sistema). Isso sem falar na confiabilidade de um *software* alterado diretamente pelo programador. Portanto, ferramentas automáticas de manutenção/reuso de *software* são de fundamental importância. ❑

5.2.4 Leitura Complementar

É importante observar que, devido a sua memória finita e predefinida, os autômatos finitos possuem limitações sérias para solucionar problemas. De fato, as linguagens que podem ser reconhecidas por um autômato finito, denominadas de *linguagens regulares*, pertencem à classe dos problemas mais simples. Considerando as linguagens regulares, o agrupamento de linguagens em classes introduzido no Capítulo 3 - Álgebra de Conjuntos (quando da introdução do Problema da Parada), a hierarquia de linguagens fica como ilustrado na Figura 5.7. Para ilustrar a limitação de capacidade de solucionar problemas, não existe autômato finito capaz de reconhecer parênteses balanceados, ou seja, um número qualquer de parênteses em um texto, eventualmente encadeados (aninhados), de tal forma a garantir que, para cada parêntese aberto (respectivamente fechado), existe um correspondente parêntese fechado (respectivamente, aberto).

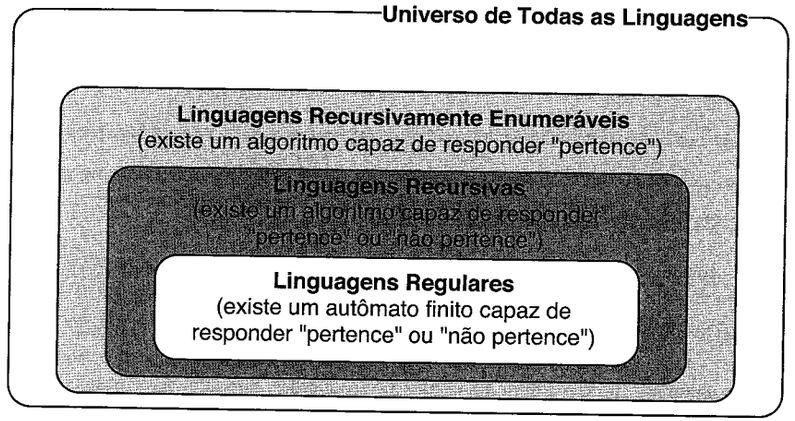


Figura 5.7 Hierarquia (continência própria) de classes de linguagens

Entretanto, relativamente à *complexidade de algoritmos*, autômato finitos pertencem à classe de algoritmos mais eficientes em termos de tempo de processamento (supondo que toda a entrada necessita ser lida). Adicionalmente, qualquer autômato que solucione um problema é igualmente eficiente. Ou seja, qualquer solução é ótima. Por fim, a implementação computacional de autômatos finitos é trivial.

5.3 Função Total

Já foi afirmado que uma função (total) é simplesmente uma função parcial a qual é total. Portanto, funções totais herdam os conceitos e terminologias introduzidos para relações e funções parciais.

Alguns resultados são discutidos no contexto particular das funções, com destaque para:

- condições para que a dual de uma função seja uma função;
- a prova de que a composição de funções é uma função;

- o fato de que os conceitos de injetora e monomorfismo (respectivamente, sobrejetora e epimorfismo) coincidem no contexto das funções.

Por fim, é introduzido o conceito de função bijetora.

5.3.1 Definição e Introdução

Definição 5.7 - Função, Aplicação
Uma *Aplicação*, *Função Total* ou simplesmente *Função* é uma função parcial $f: A \to B$ a qual é total. ❑

Portanto, uma função (total) é uma função parcial definida para todos os elementos do domínio.

EXEMPLO 5.11 - Função
Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então (procure justificar cada um dos itens):

a) São funções:

$=: A \to B$
$id_B: B \to B$
$x^2: \mathbf{Z} \to \mathbf{Z}$ tal que $x^2 = \{\langle x, y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$
$ad: \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ tal que $ad\langle a, b\rangle = a + b$
$\varnothing: \varnothing \to \varnothing$

mas tem Exemplos nesse Livro!!

b) Não são funções:

$\varnothing: A \to B$
$\{\langle 0, a\rangle, \langle 1, b\rangle\}: C \to B$
$A \times B: A \to B$
$<: C \to C$
$div: \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ tal que $div\langle x, y\rangle = x/y$ ❑

Em termos da notação como matriz ou como grafo (no caso de uma endorrelação), basta considerar que uma função é uma relação funcional e total. Assim:

- matriz: existe exatamente um valor verdadeiro em cada *linha* da matriz;
- grafo: existe exatamente uma aresta *partindo* de cada nodo.

Observe que, para funções, o conceito de injetora (respectivamente, sobrejetora) coincide com o de monomorfismo (respectivamente, epimorfismo). Lembre-se de que:

- monomorfismo é total e injetora;
- epimorfismo é funcional e sobrejetora.

Para as funções parciais, o conceito de injetora (respectivamente, sobrejetora) coincide com o de monomorfismo (respectivamente, epimorfismo)?

No contexto das funções, um isomorfismo também é denominado de *função bijetora*. Conseqüentemente, uma função bijetora é uma função injetora e sobrejetora.

EXEMPLO 5.12 - Função Injetora, Sobrejetora e Bijetora
Sejam $A = \{a\}$, $B = \{a, b\}$, $C = \{0, 1, 2\}$ e X um conjunto qualquer. Então cada uma das seguintes funções é (procure justificar cada um dos itens):

$=: A \to B$ — injetora, não-sobrejetora
$id_X: X \to X$ — bijetora

$x^2: \mathbf{Z} \to \mathbf{Z}$ onde $x^2 = \{\langle x, y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$ — não-injetora, não-sobrejetora
$ad: \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ tal que $ad(a, b) = a + b$ — epimorfismo, não-monomorfismo
$\varnothing: \varnothing \to \varnothing$ — bijetora
$\{\langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 0\rangle\}: C \to C$ — isomorfismo
$R = \{\langle x, y\rangle \mid y = \operatorname{sen} x\}$ — monomorfismo, não-epimorfismo ❑

Sugere-se como exercício verificar se a restrição de uma função é sempre uma função.

5.3.2 Exemplos Importantes de Funções

Alguns importantes exemplos de funções devem ser destacados. No que segue, lembre-se de que, se Σ representa um alfabeto, então Σ^* denota o conjunto de todas as palavras possíveis sobre Σ. Por exemplo, para $\Sigma = \{a, b\}$, vale:

$$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$$

O primeiro exemplo é o de *função constante*, ou seja, função a qual, para qualquer valor do domínio, resulta sempre no mesmo valor do contradomínio.

Definição 5.8 - Função Constante
Sejam A e B conjuntos quaisquer. Uma *função constante* em $b \in B$, usualmente denotada por:

$$const_b: A \to B$$

é tal que, para todo $a \in A$, $const_b(a) = b$ ❑

EXEMPLO 5.13 - Função Constante
Sejam $A = \{a\}$ um conjunto e $\Sigma = \{a, b\}$ um alfabeto. Então são funções constantes (procure justificar cada um dos itens):

$const_5: \mathbf{R} \to \mathbf{R}$ onde $const_5 = \{\langle x, 5\rangle \in \mathbf{R}^2\}$
$palavra_vazia: \Sigma^* \to \Sigma^*$ onde $palavra_vazia = \{\langle w, \varepsilon\rangle \in \Sigma^* \times \Sigma^*\}$
$id_A: A \to A$ (toda função identidade é uma função constante?)
$\varnothing: \varnothing \to \varnothing$ ❑

Um exemplo de função especialmente importante para Computação e Informática é o de *função concatenação*. A *concatenação* é uma operação binária, definida sobre uma linguagem, a qual associa a cada par de palavras uma palavra formada pela justaposição da primeira com a segunda. Uma concatenação é usualmente denotada pela justaposição dos símbolos que representam as palavras componentes.

Definição 5.9 - Função Concatenação
Seja $\Sigma = \{a, b\}$ um alfabeto. A *função de concatenação*:

$$conc: \Sigma^* \times \Sigma^* \to \Sigma^*$$

é tal que, para todo $\langle u, v\rangle \in \Sigma^* \times \Sigma^*$, $conc\langle u, v\rangle = u\,v$ ❑

EXEMPLO 5.14 - Concatenação
Considere o alfabeto $\Sigma = \{a, b\}$. A concatenação das palavras aba e bbb de Σ^* resulta na palavra ababbb, a qual também é palavra de Σ^*. ❑

Existem algumas funções importantes as quais são induzidas por relações ou operações sobre conjuntos, com destaque para as seguintes:

- *função inclusão*, a qual reflete a continência de conjuntos. De fato, toda continência induz um função inclusão e vice-versa;

- *função projeção*, a qual reflete a relação que existe entre o conjunto resultante de um produto cartesiano e os conjuntos originais;
- *função imersão*, a qual reflete a relação que existe entre o conjunto resultante de uma união disjunta e os conjuntos originais.

As duas últimas caracterizam a reversibilidade das operações produto cartesiano e união disjunta.

Definição 5.10 - Função Inclusão
Sejam A e B conjuntos tais que $A \subseteq B$. Uma *função inclusão*, usualmente denotada por:

$$inc_{A,B}: A \hookrightarrow B \text{ ou simplesmente } inc: A \hookrightarrow B$$

é tal que, para todo $a \in A$, $inc(a) = a$ ❑

Toda função inclusão é injetora (e portanto, monomorfismo), ou seja, $inc_{A,B}: A \rightarrowtail B$ (por quê?).

EXEMPLO 5.15 - Função Inclusão × Continência
a) Sejam $Vogais = \{a, e, i, o, u\}$ e $Letras = \{a, b, c, \ldots, z\}$ dois conjuntos. Claramente $Vogais \subseteq Letras$. Então a função inclusão abaixo é ilustrada na Figura 5.8:

$$inc_{Vogais,Letras}: Vogais \hookrightarrow Letras$$

b) As continências $\mathbf{N} \subseteq \mathbf{Z}$, $\mathbf{Z} \subseteq \mathbf{Q}$ e $\mathbf{Q} \subseteq \mathbf{R}$ induzem as seguintes funções inclusão:

$$inc_{\mathbf{N},\mathbf{Z}}: \mathbf{N} \rightarrowtail \mathbf{Z} \quad inc_{\mathbf{Z},\mathbf{Q}}: \mathbf{Z} \rightarrowtail \mathbf{Q} \quad inc_{\mathbf{Q},\mathbf{R}}: \mathbf{Q} \rightarrowtail \mathbf{R}$$

Pode-se afirmar que existe a função de inclusão $inc_{\mathbf{N},\mathbf{R}}: \mathbf{N} \rightarrowtail \mathbf{R}$? Generalizando, a composição de funções inclusão é uma função inclusão? ❑

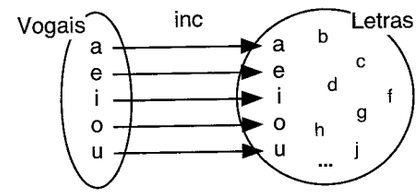


Figura 5.8 Função inclusão induzida por uma continência

EXEMPLO 5.16 - Função Inclusão × União e Intersecção
Para quaisquer conjuntos A e B, tem-se que:

$A \cap B \subseteq A$ e $A \cap B \subseteq B$
$A \subseteq A \cup B$ e $B \subseteq A \cup B$

Portanto, cada inclusão acima induz uma função inclusão as quais são ilustradas na Figura 5.9. ❑

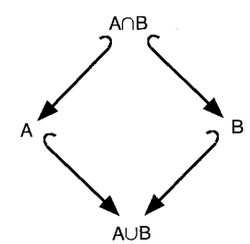


Figura 5.9 Função inclusão: união e intersecção

Já foi visto que a reversibilidade do produto cartesiano pode ser obtida como segue: o primeiro operando (respectivamente, o segundo operando) é o conjunto constituído por todos os elementos da primeira (respectivamente, da segunda) componente dos pares do produto cartesiano. Essa técnica induz duas funções projeção.

Definição 5.11 - Função Projeção

Sejam A e B conjuntos não-vazios e $A \times B$ o correspondente produto cartesiano. As *Funções Projeção*, usualmente denotadas por:

$$\pi_1 : A \times B \to A \quad \text{e} \quad \pi_2 : A \times B \to B$$

são tais que, para todo $\langle a, b \rangle \in A \times B$, vale:

$$\pi_1 \langle a, b \rangle = a \quad \text{e} \quad \pi_2 \langle a, b \rangle = b$$ ❑

Portanto, a recuperação dos operandos originais de um produto cartesiano $A \times B$ é como segue:

$A = \{ \pi_1 \langle a, b \rangle \mid \langle a, b \rangle \in A \times B \}$ primeiro operando

$B = \{ \pi_2 \langle a, b \rangle \mid \langle a, b \rangle \in A \times B \}$ segundo operando

EXEMPLO 5.17 - Função Projeção

Sejam $A = \{a, b\}$ e $B = \{x, y\}$ conjuntos e $A \times B$ o correspondente produto cartesiano. Então as funções projeção $\pi_1 : A \times B \to A$ e $\pi_2 : A \times B \to B$ são ilustradas na Figura 5.10. ❑

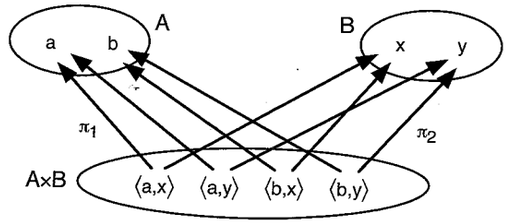


Figura 5.10 Funções projeção

Toda função projeção é sobrejetora (por quê?).

Definição 5.12 - Função Imersão

Sejam A e B conjuntos e $A + B$ a correspondente união disjunta. As *Funções Imersão*, usualmente denotadas por:

$$q_1 : A \to A + B \quad \text{e} \quad q_2 : B \to A + B$$

são tais que:

para todo $a \in A$, vale que $q_1(a) = \langle a, 0 \rangle$

para todo $b \in B$, vale que $q_2(b) = \langle b, 1 \rangle$ ❑

EXEMPLO 5.18 - Função Imersão

Sejam $A = \{a, b\}$ e $B = \{b, c\}$ conjuntos e $A + B$ a correspondente união disjunta. Então as funções de imersão $q_1 : A \to A + B$ e $q_2 : B \to A + B$ são ilustradas na Figura 5.11.

Toda função imersão é injetora (por quê?). ❑

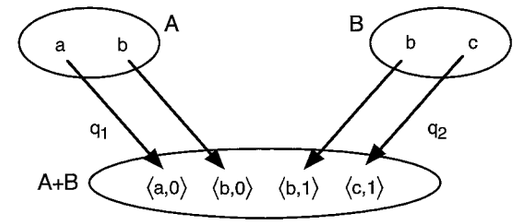


Figura 5.11 Funções imersão

5.3.3 Função Dual

A relação dual de uma função não necessariamente é uma função, como ilustrado no seguinte exemplo.

EXEMPLO 5.19 - Relação Dual de Função

a) Seja a função $f : \{0, 1, 2\} \to \{0, 1\}$ tal que $f = \{\langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 2, 0 \rangle\}$. Então a correspondente relação dual f^{op} não é função (não é funcional):

$$f^{op} = \{\langle 0, 0 \rangle, \langle 1, 1 \rangle, \langle 0, 2 \rangle\}$$

b) Seja a função $g : \{0, 1\} \to \{0, 1, 2\}$ tal que $g = \{\langle 0, 0 \rangle, \langle 1, 1 \rangle\}$. Então a correspondente relação dual não é função (por quê?):

$$g^{op} = \{\langle 0, 0 \rangle, \langle 1, 1 \rangle\}$$

Observe que os conjuntos de pares ordenados correspondentes a g e g^{op} são *iguais*, ou seja:

$$g = g^{op} \quad \text{são tais que} \quad \{\langle 0, 0 \rangle, \langle 1, 1 \rangle\} = \{\langle 0, 0 \rangle, \langle 1, 1 \rangle\}$$

Então, por que g *é* função enquanto que g^{op} *não* é função? ❑

Considerando que os conceitos duais de total e funcional são sobrejetora e injetora, respectivamente, então para que a dual de uma função (relação total e funcional) seja função, esta deve ser sobrejetora e injetora, ou seja, uma bijeção.

EXEMPLO 5.20 -Relação Dual de Função

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então:

a) As relações duais das seguintes funções são funções:

$id_B : B \to B$

$\varnothing : \varnothing \to \varnothing$

$\{\langle 0, 1 \rangle, \langle 1, 2 \rangle, \langle 2, 0 \rangle\} : C \to C$

b) As relações duais das seguintes funções *não* são funções (por quê?):

$= : A \to B$

$ad : \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ tal que $ad(a, b) = a + b$

$x^2 : \mathbf{Z} \to \mathbf{Z}$ onde $x^2 = \{\langle x, y \rangle \in \mathbf{Z}^2 \mid y = x^2\}$

$R = \{\langle x, y \rangle \mid y = \operatorname{sen} x\}$

c) Em que condições a dual de uma função inclusão é uma função? ❑

5.3.4 Composição de Funções

Já foi visto que a composição de funções parciais é uma função parcial. Então, para provar que a composição de funções é uma função, basta provar que a composição de relações totais é total.

Teorema 5.13 - Composição de Totais é Total
Sejam $R: A \to B$ e $S: B \to C$ relações totais. Então a relação composta $S \circ R: A \to C$ é uma relação total.

Prova:
Suponha que $R: A \to B$ e $S: B \to C$ são relações totais. Como a composição de relações é uma relação, então $S \circ R: A \to C$ é uma relação. Portanto, basta provar que a composição de relações totais é total, ou seja, que $S \circ R: A \to C$ é tal que:

$$(\forall a \in A)(\exists c \in C)(a\,(S \circ R)\,c)$$

De fato, suponha $a \in A$. Então:

$a \in A \Rightarrow$	R é total
$(\exists b \in B)(a\,R\,b) \Rightarrow$	S é total
$(\exists b \in B)(\exists c \in C)(a\,R\,b \wedge b\,S\,c) \Rightarrow$	definição de composição
$(\exists c \in C)(a\,(S \circ R)\,c)$	

Logo, $S \circ R: A \to C$ é uma relação total. ❑

Portanto, a composição de funções é uma função.

EXEMPLO 5.21 - Composição de Funções
Considere a Figura 5.12. A composição das funções $f: A \to B$ e $g: B \to C$ é $g \circ f: A \to C$ sendo que:

$f = \{\langle a, 1\rangle, \langle b, 2\rangle, \langle c, 5\rangle, \langle d, 5\rangle\}$
$g = \{\langle 1, x\rangle, \langle 2, y\rangle, \langle 3, y\rangle, \langle 4, y\rangle, \langle 5, z\rangle\}$
$g \circ f = \{\langle a, x\rangle, \langle b, y\rangle, \langle c, z\rangle, \langle d, z\rangle\}$ ❑

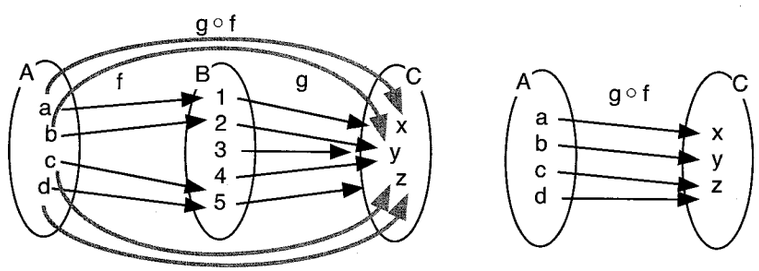


Figura 5.12 Composição de funções

Por dualidade do Teorema 5.13 - Composição de Totais é Total, a composição de relações sobrejetoras é uma relação sobrejetora. Sugere-se, como exercício, fazer a prova detalhada do seguinte corolário, "dualizando" a prova do teorema em questão.

Corolário 5.14 - Composição de Sobrejetoras é Sobrejetora
Sejam $R: A \to B$ e $S: B \to C$ relações sobrejetoras. Então a relação composta $S \circ R: A \to C$ é uma relação sobrejetora. ❑

5.4 Construções Matemáticas como Funções

Funções são freqüentemente usadas para definir outras construções matemáticas como, por exemplo, seqüência e multiconjunto. Entre as construções já estudadas, destaca-se o fato de que qualquer relação pode ser definida como uma função.

5.4.1 Relação como Função

Lembre-se de que qualquer relação $R: A \to B$ é um subconjunto do produto cartesiano $A \times B$, ou seja, é tal que $R \subseteq A \times B$. Como todo subconjunto define uma função inclusão, uma *relação* pode ser vista como segue:

$$inc_{R,A\times B}: R \rightarrowtail A \times B$$

De fato, esta é uma definição alternativa para relação. O leitor deve sempre manter em mente que, na Matemática, assim como na Computação e Informática, é usual encontrar definições alternativas equivalentes para uma mesma construção. ❑

5.4.2 Multiconjunto

Informalmente, um conjunto foi definido como sendo:

uma coleção, sem repetições e sem qualquer ordenação, de objetos denominados elementos

A definição de conjunto apresentada foi a seguinte:

uma coleção de zero ou mais objetos distintos, chamados elementos do conjunto, os quais não possuem qualquer ordem associada

Observe que uma característica fundamental de qualquer definição, formal ou informal, de conjunto é o fato de que os elementos são distintos, ou seja, não podem ser repetidos. De fato, foi mostrado que, pela definição de igualdade de conjuntos, vale, por exemplo:

$$\{1, 2, 3\} = \{3, 3, 3, 2, 2, 1\}$$

Entretanto, existem situações em que é necessário tratar conjuntos com repetições de elementos, denominados de *multiconjuntos*, como, por exemplo:

- união disjunta;
- grafos.

Na união disjunta, foi apresentada uma solução a qual garante uma identidade única de cada elemento usando uma noção de "sobrenome". Tal solução (não-baseada em multiconjuntos) é especialmente interessante, pois permite a reversibilidade da operação. Uma solução alternativa (mas que não permite a reversibilidade) seria definir o conjunto resultante da união disjunta como um multiconjunto.

A forma usual de definir multiconjunto é como uma função com origem no conjunto de objetos e destino no conjunto dos números naturais. Assim, o número natural associado a cada elemento corresponde ao número de repetições (zero ou mais) que deve ser considerado no multiconjunto.

Definição 5.15 - Multiconjunto
Suponha X um conjunto. Então um *Multiconjunto* A de objetos de X é uma função:

$$A: X \rightarrow \mathbf{N}$$ ❑

Um multiconjunto é usualmente denotado como um conjunto, explicitando as repetições, mas destacando que se trata de um multiconjunto.

EXEMPLO 5.22 - Multiconjunto
Considere a Figura 5.13. Os seguintes *multiconjuntos* são distintos:

$$\{1,2,3\} \neq \{3,3,3,2,2,1\}$$ ❑

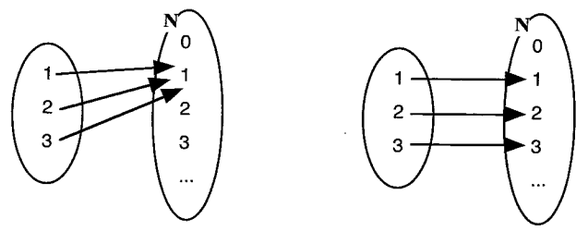


Figura 5.13 Multiconjuntos distintos

Qual a interpretação quando o número de repetições de um elemento em um multiconjunto é zero?

EXEMPLO 5.23 - Grafo com Arcos Paralelos
Considere o grafo G com arcos paralelos (possuem os mesmos nodos origem e destino) ilustrado na Figura 5.14. O correspondente multiconjunto que representa o grafo é como segue (trata-se de uma relação?):

$$G = \{\langle A, B\rangle, \langle A, B\rangle, \langle B, B\rangle, \langle B, B\rangle, \langle B, B\rangle\}$$

Sugere-se como exercício, detalhar a função que define o multiconjunto em questão. ❑

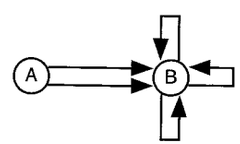


Figura 5.14 Grafo com arcos paralelos

5.4.3 Seqüência

O termo seqüência vem sendo usado intuitivamente ao longo de todo texto. Quando da definição de produto cartesiano, foi introduzida uma noção um pouco mais formal de seqüência (finita) ou de n-upla ordenada, a qual é reproduzida a seguir:

uma seqüência de n componentes, denominada de n-upla ordenada consiste de n objetos (não necessariamente distintos) em uma ordem fixa

Destaca-se que uma n-upla ordenada $\langle x_1, x_2, x_3, \ldots, x_n\rangle$ não deve ser confundida com o conjunto $\{x_1, x_2, x_3, \ldots, x_n\}$: em uma n-upla ordenada, a ordem é importante.

Uma forma simples de definir uma *seqüência finita* de n componentes é como uma função com origem no conjunto $\{1, 2, 3, \ldots, n\}$ e destino no conjunto de objetos. Assim, para $i \in \{1, 2, 3, \ldots, n\}$, a imagem de i corresponde à i-ésima componente x_i da seqüência. Generalizando, uma *seqüência infinita* teria como domínio o conjunto dos números naturais sem o zero.

Definição 5.16 - Seqüência Infinita, Seqüência Finita
Seja X um conjunto. Então:

a) Uma *Seqüência Infinita* $\underline{x}$ de objetos de X é uma função:

$$\underline{x}: \mathbf{N} - \{0\} \rightarrow X$$

a) Uma *Seqüência Finita* ou n-*Upla Ordenada* $\underline{x}$ com n componentes de objetos de X é uma função:

$$\underline{x}: \{1, 2, 3, \ldots, n\} \rightarrow X$$ ❑

EXEMPLO 5.24 - Palavra como Função
Um exemplo de seqüência finita já apresentado foi o de palavra sobre um alfabeto. Por exemplo, supondo o alfabeto $\Sigma = \{a, b, c, \ldots, z\}$, a palavra casa pode ser vista como uma função (não-injetora) como segue (veja a Figura 5.15):

$$casa: \{1, 2, 3, 4\} \rightarrow \Sigma$$

a qual é definida pelo seguinte conjunto de pares ordenados:

$$casa = \{\langle 1, c\rangle, \langle 2, a\rangle, \langle 3, s\rangle, \langle 4, a\rangle\}$$ ❑

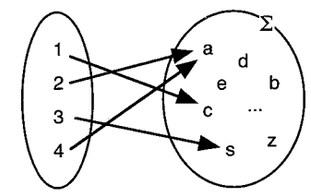


Figura 5.15 Palavra como função

5.4.4 Conjunto Indexado

Quando da introdução do conceito de variável arranjo de uma linguagem de programação como Pascal, foi afirmado que:

uma variável do tipo arranjo é uma seqüência finita de variáveis, todas do mesmo tipo

Como ilustração, considere a seguinte definição de variável do tipo arranjo:

```
dados = array[1..10] of char
```

Observe que uma variável do tipo arranjo é uma seqüência. No exemplo, a correspondente função é:

$$dados: \{1, 2, 3, \ldots, 10\} \rightarrow X$$

na qual X é um conjunto de variáveis do tipo `char` e é tal que:

a) *Injetora*: Cada componente de um arranjo é distinta. Portanto, a função é injetora, ou seja:

$$\text{dados}:\{1,2,3,\ldots,10\} \rightarrowtail X$$

Por exemplo, dados(1) e dados(8) são variáveis do tipo `char` distintas e correspondem às componentes `dados[1]` e `dados[8]`, respectivamente;

b) *Sobrejetora*. Cada componente do arranjo é indexável por algum índice. Portanto, a função é sobrejetora:

$$\text{dados}:\{1,2,3,\ldots,10\} \twoheadrightarrow X$$

Logo, a função que define uma variável do tipo arranjo como uma seqüência de variáveis é bijetora. No caso ilustrado, tem-se que:

$$\text{dados}:\{1,2,3,\ldots,10\} \leftrightarrow X$$

A generalização do raciocínio acima define o conceito de conjunto indexado.

Definição 5.17 - Conjunto Indexado

Sejam I e X conjuntos. Então, para uma dada função bijetora:

$$f: I \leftrightarrow X$$

afirma-se que X é um *Conjunto Indexado* pelo conjunto I. ❑

Portanto, qualquer função bijetora define um conjunto indexado. Para $f: I \leftrightarrow X$, I é denominado de *conjunto de índices*. Para reforçar a noção de indexação de X por I, um elemento $x \in X$ é genericamente denotado usando o seu correspondente índice $i \in I$, ou seja:

$$f(i) \text{ é denotado por } x_i$$

5.5 Função de *Hashing*

Suponha uma situação em que se deseja armazenar e recuperar informações de forma eficiente em termos de espaço de armazenamento e de tempo de recuperação. O armazenamento e a recuperação podem ser feitos em uma *tabela* (variável do tipo arranjo) ou em um *arquivo de acesso direto* (arquivo no qual cada entrada/registro pode ser diretamente acessado). Como ilustração, suponha que se trata de uma tabela.

Uma forma simples e eficiente é usar a chave de identificação (por exemplo, o número de matrícula dos alunos) como índice da tabela. Entretanto, quando o número de entradas identificáveis pela chave é muito superior ao número de entradas que serão efetivamente armazenadas (como por exemplo, o cadastro de clientes de uma loja usando o CIC como chave de identificação), resultará em um espaço de armazenamento excessivamente grande e esparso (ou seja, com baixa ocupação).

Portanto, a questão é: para uma tabela com um número restrito de entradas como, usando uma chave com capacidade de endereçamento relativamente grande, endereçar eficientemente a correspondente entrada na tabela? Ou seja, supondo que $\text{Chaves} = \{c_1, c_2, c_3, \ldots, c_m\}$ é o conjunto de todas as chaves de identificação, que n é o número de entradas na tabela e que m é possivelmente muito maior do que n, o que se deseja é uma função de endereçamento:

$$f: \text{Chaves} \rightarrow \{1,2,3,\ldots,n\}$$

A função utilizada para obter o endereço de instalação é denominada de *função de cálculo de endereço, função de aleatorização, função de randomização* ou *função de hashing*.

A função ideal é aquela que gera um endereço diferente para cada um dos possíveis valores das chaves a serem instaladas na tabela, isto é, o ideal é uma função *injetora*. No entanto, é difícil conseguir um monomorfismo, e, portanto, as funções geralmente utilizadas geram colisões, isto é, para alguns casos, podem atribuir o mesmo endereço a chaves com valores diferentes. Ou seja, para chaves de identificação $c_1 \in \text{Chaves}$ e $c_2 \in \text{Chaves}$, pode ocorrer a seguinte situação de colisão:

$$c_1 \neq c_2 \quad \text{e} \quad f(c_1) = f(c_2) \qquad \textit{colisão}$$

EXEMPLO 5.25 - Função de Hashing

Considere uma chave de identificação numérica com valores entre 0 e 1000 bem como uma tabela de armazenamento com entradas indexadas de 1 a 23. Uma função de *hashing* relativamente simples e razoavelmente eficiente é a seguinte:

$$\text{hash}:\{0,1,\ldots,1000\} \rightarrow \{1,2,\ldots,23\}$$

tal que para $c \in \{0,1,\ldots,1000\}$, tem-se que (suponha que mod calcula o resto da divisão inteira):

$$\text{hash}(c) = (c \bmod 23) + 1$$

A seguir, é apresentado um possível conjunto de valores de chaves e os correspondentes endereços calculados pela função hash:

Chave	452	623	766	564	825	387	237	360	134	285
Endereço	16	3	8	13	21	20	8	16	20	10

Observe que ocorreram colisões. De fato:

- o endereço 8 é obtido para as chaves 766 e 237;
- analogamente, os endereços 16 e 20 também são gerados para um par de chaves diferentes. ❑

Para obter o efeito de uma função injetora no cálculo do endereçamento, são utilizados métodos de tratamento de colisões. Esses métodos consistem, essencialmente, no estabelecimento de uma política para a escolha de uma entrada disponível na qual será armazenada cada chave que vier a colidir durante o processo de inserção. O detalhamento do tratamento de colisões e de outras questões correlatas é desenvolvido no estudo das *Estruturas de Dados*.

5.6 Funções nas Linguagens de Programação

A grande maioria das linguagens de programação é capaz de manipular construções similares ou baseadas nas funções matemáticas. Algumas diferenças de conceitos (função matemática comparativamente com funções em linguagens de programação) são detalhadas na seção 5.7 - Linguagem de Programação Funcional. No caso da linguagem Pascal, existe a declaração `function`, a qual é apresentada via exemplos.

EXEMPLO 5.26 - Função em Pascal: Hashing

Considere o EXEMPLO 5.25 - Função de *Hashing* e suponha a declaração de dois tipos intervalos como segue:

```
type  interv_0_1000 = 0..1000;
      interv_1_23 = 1..23
```

Então, uma declaração da função de *hashing* $f: \{0, 1, \ldots, 1000\} \rightarrow \{1, 2, \ldots, 23\}$ tal que $f(c) = (c \bmod 23) + 1$ é como segue:

```
function hash(c: interv_0_1000): interv_1_23;
    begin
    hash := (c mod 23) + 1
    end
```

Nessa declaração de `function`, observa-se que:

- o parâmetro `c` do tipo `interv_0_1000` define o domínio da função e é denominado de *parâmetro formal*;
- `hash`, do tipo `interv_1_23` define o contradomínio da função e conterá o valor resultante do cálculo quando da chamada da função, como ilustrado a seguir:

```
if    hash(766) = hash(237)
then  ...
```

Nas chamadas da função `hash` acima, os valores `766` e `237` são denominados de *parâmetros atuais*. Nesse caso, o comando após a palavra `then` é executado, pois:

$$\text{hash}(766) = \text{hash}(237) = 8$$

❑

EXEMPLO 5.27 - Função em Pascal: EXOR

Um conetivo lógico muito comum em Computação e Informática é o *Ou-Exclusivo*, usualmente denominado de EXOR (do inglês, *exclusive or*), cuja semântica é dada pela tabela verdade ilustrada na Figura 5.16. O conetivo EXOR pode ser reescrito, usando os conetivos usuais, como segue:

$$p \text{ EXOR } q \Leftrightarrow (p \wedge \neg q) \vee (\neg p \wedge q)$$

Um exemplo de função em Pascal que implementa o conetivo EXOR é como segue:

```
function exor(p, q: boolean): boolean;
    begin
    exor := (p and not q) or (not p and q)
    end
```

❑

p	q	p EXOR q
V	V	F
V	F	V
F	V	V
F	F	F

Figura 5.16 Tabela verdade: EXOR

Um problema das funções em Pascal é que estas não permitem um contradomínio resultante de um produto cartesiano. Por exemplo, suponha a equação polinomial do segundo grau do tipo:

$$a x^2 + b x + c = 0$$

Uma solução, usando a fórmula de Baskara, resulta em duas raízes, ou seja, é uma função do tipo:

$$\text{baskara}: \mathbf{R}^3 \rightarrow \mathbf{R}^2$$

Tal função, embora implementável em Pascal usando alguns artifícios, não corresponde ao conceito matemático de função. Na seção seguinte, é discutida uma solução seguindo o conceito de função matemática e usando linguagem de programação funcional.

5.7 Linguagem de Programação Funcional

Programação funcional é um estilo de programação baseada em funções e composição de funções (constituindo um programa) em vez de comandos. Assim, um programa é uma expressão funcional a qual é avaliada, e não uma seqüência de comandos a serem executados por um computador (como em Pascal, por exemplo).

Uma *linguagem de programação funcional* é uma linguagem que suporta e encoraja esse estilo de programação. Assim, resumidamente, uma linguagem de programação funcional pode ser considerada como operações tipadas (ou seja, com tipos associados) e construtores que permitem definir novos tipos e operações mais complexos. Inclusive, uma linguagem de programação funcional considerada *pura* não possui variáveis, nem atribuições. Portanto, uma linguagem de programação funcional é composta basicamente por:

- tipos primitivos de dados da linguagem;
- constantes de cada tipo primitivo de dado;
- operações as quais são funções sobre os tipos primitivos de dados da linguagem;
- construtores que permitem definir novos tipos e operações derivados dos tipos e operações da linguagem.

5.7.1 Haskell

Haskell é um exemplo de linguagem de programação funcional pura na qual todas as funções devem seguir o conceito matemático de função, isto é, os mesmo valores do domínio (parâmetros atuais) aplicados à função resultam nos mesmos elementos do contradomínio (produz as mesmas saídas). Ou seja, o resultado da aplicação de uma função é independente de qualquer estado implícito do sistema. Por exemplo, considere o seguinte exemplo de função constante em Haskell:

```
x = 1
```

Trata-se de uma função constante a qual sempre retorna o valor `1`. Considere também a seguinte função:

```
f x = x + 1
```

Trata-se de uma função `f` que, para o parâmetro `x`, retorna o valor `x + 1`. Ou seja, são exemplos de funções em Haskell que satisfazem o conceito de função matemática. Um contra-exemplo em Pascal é o seguinte (suponha que `x` é uma variável do tipo `integer`):

```
function contador: integer;
    begin
    x := x + 1;
    contador := x
    end
```

A função em Pascal `contador` *não* é uma função matemática, pois cada vez que é referenciada no programa, o valor retornado é diferente (observe que `x` é uma variável definida externamente à função e, portanto, cada vez que a função contador é referenciada, é executada para valores possivelmente *diferentes* de `x`).

Em Haskell, no exemplo acima, necessariamente `x` teria que ser um parâmetro explícito de `contador` (ou utilizar recursos mais elaborados tais como *mônada*, conceito inspirado em *Teoria das Categorias* o qual está fora do escopo desta introdução). Observe-se que algumas linguagens funcionais, como *SML* ou *Scheme*, incorporam a noção de estado e permitiriam esse tipo de código.

Considere o seguinte exemplo (menos trivial) de função em Haskell para calcular as raízes de uma equação polinomial de segundo grau $a x^2 + b x + c = 0$

```
baskara a b c =
let delta = b*b - 4*a*c
in    ( (-b + sqrt(delta))/(2*a),
        (-b - sqrt(delta))/(2*a) )
```

Trata-se de uma função que, para `a`, `b` e `c`, retorna o par ordenado de valores correspondendo às raízes da equação. A palavra chave `let` é usada para declarar `delta` a qual fica acessível apenas dentro do escopo da função `baskara`. Por exemplo, aplicando os valores `1`, `-5` e `6`, a expressão seria reduzida (avaliada) pelo computador de forma similar à seguinte seqüência:

```
baskara 1 -5 6
```

```
let delta = (-5)*(-5) - 4*1*6
in    ( (5 + sqrt(delta))/(2*1),
        (5 - sqrt(delta))/(2*1) )
```

```
let delta = 1
in    ( (5 + sqrt(delta))/2,
        (5 - sqrt(delta))/2 )
```

```
( (5 + 1)/2,
(5 - 1)/2 )
```

```
( 3, 2 )
```

5.7.2 Leitura Complementar

Uma característica de que a maioria das linguagens funcionais dispõe é um sistema de tipos estático que não requer declaração de tipos na maioria das expressões. O sistema de tipos tenta inferir o tipo dos dados pelas operações (algumas expressões podem ser ambíguas e exigir que o programador declare seus tipos). Por exemplo, dada a seguinte expressão:

```
incrementa x = x + 1
```


o sistema de tipos automaticamente infere que o correspondente tipo é:

```
integer -> integer
```

ou seja, uma endofunção nos inteiros. Entretanto, se, por exemplo, a função é referenciada como segue:

```
incrementa 2.1
```

o sistema percebe que o tipo inteiro não é aplicável e modifica a inferência.

Uma característica incomum que se encontra, por exemplo, em Haskell, é a *avaliação procrastinada* (*lazy evaluation*), a qual é realizada de fora para dentro. Por exemplo, feita a seguinte declaração:

```
f x = 1
```

e a chamada:

```
f(1/0)
```

na maioria das linguagens de programação, primeiro é avaliada a divisão `1/0`, retornando uma mensagem de erro tal como:

Erro: divisão por zero

A mesma chamada em Haskell retorna o valor `1`. Uma importante conseqüência dessa característica é possibilidade de manipular estruturas de dados possivelmente infinitas. Por exemplo:

```
[1.. ]
```

é uma lista infinita cujo valor é `[1, 2, 3, ..]`. No entanto, a chamada da função:

```
pegue_os_primeiros_3 [1.. ]
```

retorna `[1,2,3]`.

Linguagens funcionais são menos utilizadas que *linguagens imperativas*, como, por exemplo, Pascal, C, etc., mas existem duas aplicações bem populares cujo estilo de programação é próximo: planilhas eletrônicas e linguagens de consultas de banco de dados (embora estas sejam mais relacionais do que funcionais).

Entretanto, em muitos cursos superiores como Engenharia, Física e ciências exatas em geral, com freqüência (principalmente na Europa) a primeira linguagem de programação é uma linguagem funcional. Destacam-se duas razões para tanto:

- cálculos complexos envolvendo o conceito de função são freqüentes e importantes;
- nas primeiras aulas, o aluno é capaz de implementar funções com razoável nível de complexidade, pois a programação é muito próxima daquela que ele desenvolve nos seus estudos. Comparativamente com o ensino de linguagem imperativas (como Pascal), no final de uma disciplina típica, o aluno ainda tem dificuldades de implementar funções mais complexas.

Mais recentemente, uma das áreas em que as linguagens de programação funcional estão se tornando populares é na manipulação de *XML*. De fato, a linguagem padrão para transformação de documentos XML é funcional, e muitas das linguagens de consulta XML propostas como padrões são funcionais.

5.8 Exercícios

Exercício 5.1 Sejam $A=\{a\}$, $B=\{a,b\}$ e $C=\{0,1,2\}$.

a) Para cada item abaixo, faça o seguinte:

- justifique por que *são* funções parciais;
- determine o domínio de definição e o conjunto imagem;
- determine o tipo da função (injetora, monomorfismo, etc.):

a.1) $\varnothing: A \to B$
a.2) $\{\langle 0,a\rangle, \langle 1,b\rangle\}: C \to B$
a.3) $=: A \to B$
a.4) $x^2: \mathbf{Z} \to \mathbf{Z}$ tal que $x^2 = \{\langle x,y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$
a.5) $ad: \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ tal que $ad\langle a,b\rangle = a+b$
a.6) $div: \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ tal que $div\langle x,y\rangle = x/y$

b) Justifique por que *não* são funções parciais:

b.1) $A \times B: A \to B$
b.2) $<: C \to C$

Exercício 5.2 Sobre a dualidade de uma função parcial:

a) Por que a dual da seguinte função parcial *não* é uma função parcial?

$$x^2: \mathbf{Z} \to \mathbf{Z} \text{ onde } x^2 = \{\langle x,y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$$

b) As duais das seguintes funções parciais são funções parciais? Justifique a sua resposta:

b.1) $ad: \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ tal que $ad\langle a,b\rangle = a+b$
b.2) $div: \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ tal que $div\langle x,y\rangle = x/y$

Exercício 5.3 Por que as seguintes funções parciais não são componíveis?

$ad: \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ tal que $ad\langle a,b\rangle = a+b$
$div: \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ tal que $div\langle x,y\rangle = x \text{ DIV } y$ sendo que DIV é a divisão inteira, desprezando o resto (exemplo: $7 \text{ DIV } 3 = 2$)

Exercício 5.4 Sejam $A=\{a\}$, $B=\{a,b\}$ e $C=\{0,1,2\}$. Usando diagramas de Venn, determine todas as composições possíveis entre as seguintes funções parciais:

$\varnothing: A \to C$
$\{\langle 0,a\rangle, \langle 1,b\rangle\}: C \to B$
$=: B \to A$

Exercício 5.5 "Dualizando" a prova do Teorema 5.2 - Composição de Funcionais é Funcional, desenvolva a prova detalhada do Corolário 5.4 - Composição de Injetoras é Injetora.

Exercício 5.6 A operação de restrição foi definida sobre o domínio de uma função parcial.

a) Como seria a operação de *restrição do contradomínio* restrição definida sobre o contradomínio?

b) Nesse caso, a relação resultante é uma função parcial?

Exercício 5.7 Para as funções parciais (justifique a sua resposta):

a) O conceito de injetora coincide com o de monomorfismo?

b) O conceito de sobrejetora coincide com o de epimorfismo?

Exercício 5.8 Relativamente a autômatos finitos, suponha um alfabeto $\Sigma=\{a,b\}$, desenvolva uma função programa (e identifique o estado inicial e os finais) de tal forma a reconhecer as seguintes linguagens sobre Σ (e rejeitar o seu complemento):

a) o conjunto de todas as palavras que possuem aaaa como subpalavra;

b) o conjunto de todas as palavras que possuem aa como sufixo;

b) o conjunto de todas as palavras que possuem um número ímpar de a e b.

Exercício 5.9 Detalhe a função programa do autômato finito resultante da operação de restrição ilustrado na Figura 5.6 (direita).

Exercício 5.10 Considere o autômato finito ilustrado na Figura 5.5. Calcule a função programa restrita ao alfabeto $\Sigma_0=\{a\}$.

Exercício 5.11 Em que condições o conjunto vazio é:

a) Uma função parcial?

b) Uma função?

Exercício 5.12 Verifique a existência de funções entre os conjuntos de cada item abaixo. No caso de existir, quantas e quais funções distintas podem ser definidas?

a) Domínio $\{a\}$ e codomínio $\{x,y,z\}$
b) Domínio $\varnothing$ e codomínio $\{x,y,z\}$
c) Domínio $\{a,b,c\}$ e codomínio $\{x\}$
d) Domínio $\{a,b,c\}$ e codomínio $\varnothing$
e) Domínio $\{a,b\}$ e codomínio $\{a,b\}$

Exercício 5.13 Sejam $A=\{a\}$, $B=\{a,b\}$ e $C=\{0,1,2\}$.

a) Para cada item abaixo, faça o seguinte:

- justifique por que *são* funções;
- determine o domínio de definição e o conjunto imagem;
- determine o tipo da função (injetora, monomorfismo, etc.).

a.1) $=: A \to B$
a.2) $id_B: B \to B$
a.3) $x^2: \mathbf{Z} \to \mathbf{Z}$ tal que $x^2 = \{\langle x,y\rangle \in \mathbf{Z}^2 \mid y = x^2\}$
a.4) $ad: \mathbf{N} \times \mathbf{N} \to \mathbf{N}$ tal que $ad\langle a,b\rangle = a+b$
a.5) $\varnothing: \varnothing \to \varnothing$
a.6) $R = \{\langle x,y\rangle \in \mathbf{R} \times \mathbf{R} \mid y = \text{sen}\, x\}$

b) Justifique por que *não* são funções:

b.1) $\varnothing: A \to B$
b.2) $\{\langle 0,a\rangle, \langle 1,b\rangle\}: C \to B$
b.3) $A \times B: A \to B$
b.4) $<: C \to C$
b.5) $div: \mathbf{R} \times \mathbf{R} \to \mathbf{R}$ tal que $div\langle x,y\rangle = x/y$

Exercício 5.14 Determine todas as composições de funções que podem ser definidas sobre as seguintes funções:

a) Todas as funções com domínio e codomínio em $\{a,b\}$;

b) Todas as funções com domínio em $\{a,b,c\}$ e codomínio em $\{x\}$.

Exercício 5.15 Reescreva as seguintes funções com domínio e codomínio em **R** em termos de composição de funções elementares:

a) $f_1(x) = x + x^2$

b) $f_2(x) = (x \operatorname{sen} x + \cos x)^2$

Exercício 5.16 Sobre restrição de funções:

a) Uma restrição de uma função é sempre uma função? Justifique a sua resposta?

b) Como seria a operação de restrição definida sobre o contradomínio para uma função?

Exercício 5.17 Sobre função constante:

a) Toda função identidade é uma função constante?

b) Em que casos uma função constante é:

b.1) Injetora?

b.2) Sobrejetora?

Exercício 5.18 Sobre função inclusão:

a) Por que toda função inclusão é injetora?

b) A composição de funções inclusão é uma função inclusão? Justifique a sua resposta.

Exercício 5.19 Sobre função projeção:

a) Por que toda função projeção é sobrejetora?

b) Uma função de projeção pode ser bijetora?

Exercício 5.20 Sobre função imersão:

a) Por que toda função imersão é injetora?

b) Em que condições uma função de imersão é bijetora?

Exercício 5.21 Em que condições a dual de uma função inclusão é uma função?

Exercício 5.22 Seja a função $f: \{0, 1\} \rightarrow \{0, 1, 2\}$ tal que $f = \{\langle 0, 0\rangle, \langle 1, 1\rangle\}$. Afirma-se que a relação dual não é função:

$$f^{op} = \{\langle 0, 0\rangle, \langle 1, 1\rangle\}$$

Observe que os conjuntos de pares ordenados correspondentes a f e f^{op} são *iguais*:

$$f = f^{op} \quad \text{ou seja} \quad \{\langle 0, 0\rangle, \langle 1, 1\rangle\} = \{\langle 0, 0\rangle, \langle 1, 1\rangle\}$$

Então, por que f *é* função enquanto que f^{op} *não* é função?

Exercício 5.23 "Dualizando" a prova do Teorema 5.13 - Composição de Totais é Total, desenvolva a prova detalhada do Corolário 5.14 - Composição de Sobrejetoras é Sobrejetora.

Exercício 5.24 Na definição de multiconjunto como uma função, qual a interpretação da situação em que o número de repetições de um elemento em um multiconjunto é zero?

Exercício 5.25 Justifique ou refute: a estrutura

$$\{a, b, b, c, c, c, c, c, c, \ldots\}$$

na qual c ocorre infinitas vezes, é multiconjunto.

Exercício 5.26 Considere o grafo G com arcos paralelos ilustrado na Figura 5.14. Detalhe a função que define o multiconjunto correspondente aos arcos.

Exercício 5.27 Sejam x e y elementos distintos de um conjunto. Usando a definição de seqüência como função, mostre que os pares ordenados $\langle x, y\rangle$ e $\langle y, x\rangle$ são distintos.

Exercício 5.28 Considere a definição de seqüência como função. Para um dado alfabeto Σ, o que seria o conjunto Σ* de todas as palavras sobre Σ?

Exercício 5.29 Relativamente à linguagem Pascal, considere a definição de variável do tipo arranjo abaixo. Detalhe tal definição como um conjunto indexado.

```
ocorrencia_letras = array[a..z] of integer
```

Exercício 5.30 No EXEMPLO 5.25 - Função de *Hashing*, o divisor da função de *hashing* é o número primo 23. De fato, em geral, é escolhido um número primo como divisor. Você conseguiria inferir qual a razão de usar números primos? Caso contrário, sugere-se uma pesquisa sobre o assunto.

Exercício 5.31 Foi visto que, em geral, uma função de *hashing* não é injetora. Uma solução para obter o efeito de uma função injetora é realizar uma pesquisa seqüencial procurando a próxima entrada livre na tabela. Esboce um trecho de programa em Pascal (ou em uma linguagem de seu conhecimento) que implemente tal política de tratamento de colisões.

Dica: lembre-se de que a tabela é finita. Como pode ser tratada a questão da próxima entrada livre quando o fim da tabela for atingido?

Exercício 5.32 Um conetivo lógico muito comum em Computação e Informática é o NAND (abreviatura dos termos em inglês *not* e *and*), cuja semântica é dada pela tabela verdade ilustrada na Figura 5.16. Uma importante conseqüência da aplicação desse conetivo é que todos os demais conetivos estudados podem ser definidos usando exclusivamente o conetivo NAND. Nesse contexto:

a) Como o conetivo NAND pode ser expresso em termos dos conetivos estudados?

b) Desenvolva uma função em Pascal (ou em uma linguagem de programação de seu conhecimento) que implemente o conetivo NAND.

x	y	x NAND y
V	V	F
V	F	V
F	V	V
F	F	V

Figura 5.17 Tabela verdade: NAND

Exercício 5.33 Considere a seguinte equação polinomial de primeiro grau e o correspondente cálculo da raiz:

$$a x + b = 0$$
$$x = b/a$$

Implemente o cálculo da raiz como uma função em:

a) Pascal;

b) Haskell.

Exercício 5.34 Implemente em Pascal duas funções, uma para cada raiz da equação polinomial do segundo grau, usando a fórmula de Baskara.

Exercício 5.35 Além da operação de composição de funções (parciais ou totais), é possível definir um conjunto de operações sobre funções, constituindo uma *álgebra de funções*. No que

segue, lembre-se de que toda função (parcial ou total) é um caso particular de uma relação e, portanto, é um conjunto.

a) Sejam $f: A \to B$ e $g: X \to Y$ funções parciais quaisquer. Então, verifique se são funções parciais:

   a.1) *União* de f e g

   a.2) *Intersecção* de f e g

   Adicionalmente, defina e exemplifique:

   a.3) A função parcial $f \times g: A \times X \to B \times Y$ induzida pelo produto cartesiano dos domínios e dos contradomínios das funções f e g, denominada de *produto de funções parciais*;

b) Sejam $f: A \to B$ e $g: X \to Y$ funções totais quaisquer. Então, verifique se são funções totais:

   b.1) *União* de f e g

   b.2) *Intersecção* de f e g

   Adicionalmente, defina e exemplifique:

   b.3) A função total $f \times g: A \times X \to B \times Y$ induzida pelo produto cartesiano dos domínios e dos contradomínios das funções f e g, denominada de *produto de funções*.

6 Endorrelações, Ordenação e Equivalência

Como já introduzido, as endorrelações são especialmente importantes, razão pela qual uma série de estudos é desenvolvida especialmente para este tipo de relações. Neste capítulo, os seguintes estudos são desenvolvidos:

- *propriedades*: principais propriedades das endorrelações;
- *fecho*: extensão de uma endorrelação de forma a satisfazer determinadas propriedades;
- *ordem*: endorrelações as quais refletem uma noção intuitiva de ordem;
- *equivalência*: endorrelações as quais refletem uma noção de igualdade semântica.

Em particular, no estudo das endorrelações de ordem, são exploradas duas importantes aplicações em Computação e Informática:

- *classificação de dados*;
- *semântica de sistemas concorrentes*.

6.1 Propriedades de uma Endorrelação

Para facilitar o entendimento, as principais propriedades são inicialmente introduzidas informalmente:

a) *Reflexiva* é uma relação na qual todo elemento está relacionado consigo mesmo. Por exemplo, na relação "igualdade" sobre os números reais, claramente todo número é igual a si mesmo;

b) *Simétrica*. Na relação "parentesco", se João é parente de José (por exemplo, são irmãos), então a vice-versa também é verdadeira, ou seja, José é parente de João. Assim, simetria significa que, toda vez que um elemento estiver relacionado com outro, a vice-versa também estará relacionada;

c) *Transitiva*. Na relação "menor" sobre os números naturais, sabe-se que, caso um número seja menor que outro o qual, por sua vez, é menor que um terceiro, então o primeiro é menor que o terceiro, o que caracteriza a noção de transitividade. Já a relação "faz fronteira com" nos países na América do Sul é não-transitiva. Por exemplo, o Brasil faz fronteira com a Argentina o qual, por sua vez, faz fronteira com o Chile. Entretanto, o Brasil *não* faz fronteira com o Chile (esta não é uma relação transitiva).

Relacionadas com as propriedades reflexiva e simétrica, existem as propriedades irreflexiva e anti-simétrica. Entretanto, deve ficar bem claro que irreflexividade e anti-simetria *não* são as correspondentes noções complementares.

A representação das relações via grafos ou matrizes auxilia no entendimento e no estudo das propriedades.

Definição 6.1 - Relação Reflexiva, Relação Irreflexiva

Sejam A um conjunto e R uma endorrelação em A. Então R é uma:

a) *Relação Reflexiva*, se $(\forall a \in A)(a\,R\,a)$

b) *Relação Irreflexiva* ou *Relação Anti-Reflexiva*, se $(\forall a \in A)(\neg(a\,R\,a))$ ❑

Observe que reflexiva e irreflexiva *não* são noções complementares. De fato, a negação da propriedade reflexiva seria $(\exists a \in A)(\neg(a\,R\,a))$ (compare com a definição de irreflexiva). Inclusive, é possível definir uma relação que é:

- *simultaneamente* reflexiva e irreflexiva;
- *não* é reflexiva *nem* irreflexiva.

EXEMPLO 6.1 - Relação Reflexiva, Relação Irreflexiva
Seja $A = \{0, 1, 2\}$. As seguintes relações são:

a) Reflexivas, mas não irreflexivas:

$\langle \mathbf{N}, \leq \rangle$
$\langle \mathbf{P}(A), \subseteq \rangle$
$A^2 : A \to A$
$\langle A, = \rangle$

b) Irreflexivas, mas não reflexivas:

$\langle \mathbf{Z}, \neq \rangle$
$\langle \mathbf{P}(A), \subset \rangle$
$\varnothing : A \to A$
$\langle A, R \rangle$, supondo $R = \{\langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 1\rangle\}$

c) Nem reflexiva, nem irreflexiva:

$\langle A, S \rangle$, supondo $S = \{\langle 0, 2\rangle, \langle 2, 0\rangle, \langle 2, 2\rangle\}$ ❑

Para um conjunto finito A, uma forma simples de entender e verificar se uma endorrelação $R : A \to A$ é reflexiva ou irreflexiva, é analisar a sua representação como matriz ou como grafo, como segue:

a) *Matriz*

Reflexiva: a diagonal da matriz contém somente o valor verdadeiro;
Irreflexiva: a diagonal da matriz contém somente o valor falso;

b) *Grafo*

Reflexiva: qualquer nodo tem um arco com origem e destino nele mesmo;
Irreflexiva: qualquer nodo *não* tem um arco com origem e destino nele mesmo.

EXEMPLO 6.2 - Relação Reflexiva, Relação Irreflexiva
Seja $A = \{0, 1, 2\}$. Observe as seguintes relações as quais são ilustradas como segue (sempre da esquerda para a direita, respectivamente):

- na Figura 6.1 como matrizes, nas quais a diagonal é destacada;
- na Figura 6.2 como grafos.

a) Reflexivas, mas não irreflexivas

$A^2 : A \to A$
$\langle A, = \rangle$, dado que $=$ é definida por $\{\langle 0, 0\rangle, \langle 1, 1\rangle, \langle 2, 2\rangle\}$

b) Irreflexivas, mas não reflexivas

$\varnothing : A \to A$
$R = \{\langle 0, 1\rangle, \langle 1, 2\rangle, \langle 2, 1\rangle\}$

Como seria a matriz ou o grafo de uma relação que *não* é reflexiva *nem* irreflexiva? ❑

A^2	0	1	2
0	1	1	1
1	1	1	1
2	1	1	1

$=$	0	1	2
0	1	0	0
1	0	1	0
2	0	0	1

$\varnothing$	0	1	2
0	0	0	0
1	0	0	0
2	0	0	0

R	0	1	2
0	0	1	0
1	0	0	1
2	0	1	0

Figura 6.1 Endorrelações reflexivas/irreflexivas como matrizes

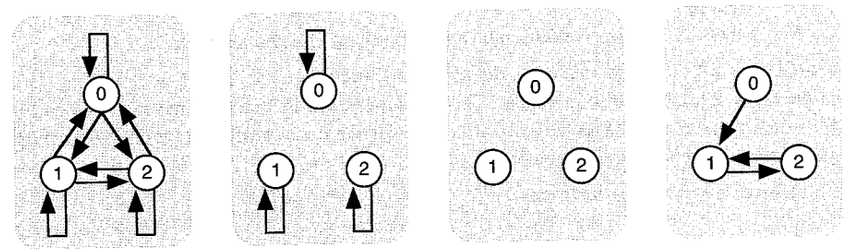


Figura 6.2 Endorrelações reflexivas/irreflexivas como grafos

Definição 6.2 - Relação Simétrica, Relação Anti-Simétrica
Sejam A um conjunto e R uma endorrelação em A. Então R é uma:

a) *Relação Simétrica*, se $(\forall a \in A)(\forall b \in A)(a\,R\,b \to b\,R\,a)$
b) *Relação Anti-Simétrica*, se $(\forall a \in A)(\forall b \in A)(a\,R\,b \wedge b\,R\,a \to a = b)$ ❑

Anti-simetria estabelece que, para uma relação R e um par $\langle x, y\rangle \in R$, não é possível inverter a ordem dos elementos (ou seja, o par $\langle y, x\rangle \notin R$), excetuando-se no caso em que x e y são iguais.

Observe que simetria e anti-simetria *não* são noções complementares. De fato, é possível definir uma relação que é simultaneamente simétrica e anti-simétrica.

EXEMPLO 6.3 - Relação Simétrica, Relação Anti-Simétrica
Seja X um conjunto qualquer. Então, as seguintes relações são:

a) Simétricas

$X^2 : X \to X$
$\langle X, = \rangle$
$\langle X, \neq \rangle$
$\langle \mathbf{P}(X), = \rangle$
$\varnothing : X \to X$

b) Anti-simétricas

$\langle X, = \rangle$
$\langle \mathbf{P}(X), = \rangle$
$\varnothing : X \to X$
$\langle \mathbf{N}, R \rangle$, supondo $R = \{\langle x, y\rangle \in \mathbf{N}^2 \mid y = x^2\}$

c) Nem simétrica, nem anti-simétrica

$S = \{\langle 0, 1\rangle, \langle 1, 0\rangle, \langle 1, 2\rangle\}$ ❑

Para um conjunto finito A, uma forma simples de entender e verificar se uma endorrelação $R : A \to A$ é simétrica ou anti-simétrica, é analisar a sua representação como matriz ou como grafo, como segue:

a) *Matriz*

Simétrica: a metade acima da diagonal da matriz é a *imagem espelhada* da metade abaixo;
Anti-simétrica: para qualquer célula verdadeira em uma das metades da matriz (em relação à diagonal), a correspondente célula na outra metade é falsa;

b) *Grafo*

Simétrica: entre dois nodos quaisquer, ou *não* existe seta, ou existem *duas* setas, uma em cada sentido;
Anti-simétrica: entre dois nodos quaisquer, existe no *máximo uma* seta.

EXEMPLO 6.4 - Relação Simétrica, Relação Anti-Simétrica

Seja $A=\{0,1,2\}$. Observe as seguintes relações as quais são ilustradas como matrizes na Figura 6.3, na qual a metade acima da diagonal está destacada, e como grafos na Figura 6.4 (sempre da esquerda para a direita, respectivamente):

a) Simétrica, mas não anti-simétrica

A^2

b) Simétrica e anti-simétrica

$\langle A, = \rangle$

c) Anti-simétrica, mas não simétrica

$R: A \to A$ tal que $R = \{\langle 0,0\rangle, \langle 1,1\rangle, \langle 1,2\rangle\}$

d) Nem simétrica, nem anti-simétrica

$S: A \to A$ tal que $S = \{\langle 0,1\rangle, \langle 1,0\rangle, \langle 1,2\rangle\}$ ❑

A^2	0	1	2
0	1	1	1
1	1	1	1
2	1	1	1

=	0	1	2
0	1	0	0
1	0	1	0
2	0	0	1

R	0	1	2
0	1	0	0
1	0	1	1
2	0	0	0

S	0	1	2
0	0	1	0
1	1	0	1
2	0	0	0

Figura 6.3 Endorrelações simétricas/anti-simétricas como matrizes

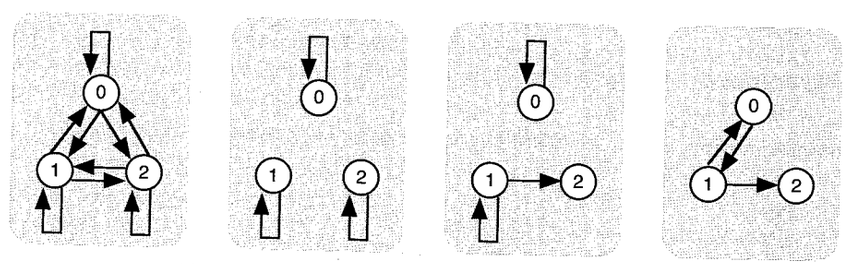


Figura 6.4 Endorrelações simétricas/anti-simétricas como grafos

Definição 6.3 - Relação Transitiva

Sejam A um conjunto e R uma endorrelação em A. Então R é uma *Relação Transitiva*, se:

$$(\forall a \in A)(\forall b \in A)(\forall c \in A)(a\,R\,b \wedge b\,R\,c \to a\,R\,c)$$ ❑

EXEMPLO 6.5 - Relação Transitiva

Sejam $A=\{0,1,2\}$ e X um conjunto qualquer. Então, as seguintes relações são:

a) Transitivas

$X^2: X \to X$
$\varnothing: X \to X$
$\langle X, = \rangle$
$\langle \mathbf{N}, \leq \rangle$
$\langle \mathbf{Z}, < \rangle$
$\langle \mathbf{P}(X), \subseteq \rangle$
$\langle \mathbf{P}(X), \subset \rangle$

b) Não-transitivas

$\langle \mathbf{Z}, \neq \rangle$ (por quê?)
$\langle A, R \rangle$, supondo $R = \{\langle 0,1\rangle, \langle 1,2\rangle, \langle 2,1\rangle\}$
$\langle A, S \rangle$, supondo $S = \{\langle 0,2\rangle, \langle 2,0\rangle, \langle 2,2\rangle\}$ ❑

Ao contrário das propriedades já introduzidas, o entendimento e visualização de uma endorrelação transitiva (sobre um conjunto finito) como matriz não é especialmente vantajoso. Entretanto, como grafo, a transitividade possui uma importante interpretação: o grafo explicita todos os *caminhos* possíveis entre dois nodos. Informalmente, um caminho em um grafo é determinado por uma seta ou por uma seqüência de setas encadeadas. Assim, se existe uma forma de ligar dois nodos via setas encadeadas, então existe um caminho entre estes nodos.

EXEMPLO 6.6 - Relação Transitiva

Seja $A=\{0,1,2\}$. Observe as seguintes relações as quais são ilustradas como grafos na Figura 6.5 (da esquerda para a direita, respectivamente):

$A^2: A \to A$
$\langle A, = \rangle$
$\langle A, \leq \rangle$
$\langle A, < \rangle$ ❑

6.2 Fecho de uma Endorrelação

Freqüentemente é desejável estender uma relação de forma a garantir que ela satisfaz determinado conjunto de propriedades. Por exemplo, garantir que determinada relação R satisfaz à propriedade reflexiva. Assim, se R não é reflexiva, são introduzidos os pares (e somente estes) que garantem a reflexividade de R.

Definição 6.4 - Fecho de uma Relação

Sejam $R: A \to A$ uma endorrelação e P um conjunto de propriedades. Então o *Fecho de* R *em Relação a* P, é a menor endorrelação em A que contém R e que satisfaz as propriedades de P e é denotado por:

$$\text{FECHO-P}(R)$$ ❑

Portanto, para qualquer conjunto de propriedades considerado, a relação sempre será subconjunto de seu fecho, ou seja:

$$R \subseteq \text{FECHO-P}(R)$$

Em que condições será igual ao seu fecho, ou seja, $R = \text{FECHO-P}(R)$?

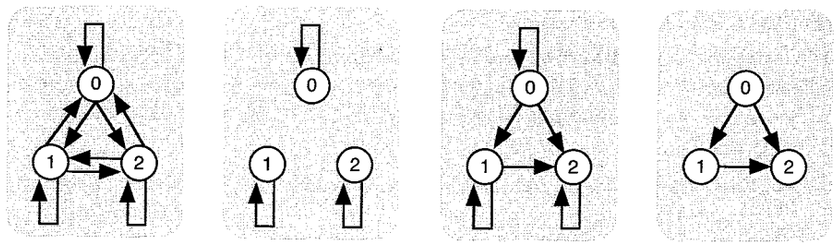


Figura 6.5 Endorrelações transitivas como grafos

Suponha uma endorrelação $R: A \rightarrow A$. A construção do fecho de R para as propriedades reflexiva, simétrica e transitiva é como segue:

a) *Fecho Reflexivo.*

$$\text{FECHO-}\{\text{reflexiva}\}(R) = R \cup \{\langle a, a\rangle \mid a \in A\}$$

b) *Fecho Simétrico.*

$$\text{FECHO-}\{\text{simétrica}\}(R) = R \cup \{\langle b, a\rangle \mid \langle a, b\rangle \in R\}$$

c) *Fecho Transitivo.* O FECHO-{transitiva}(R) é construído como segue:

c.1) Se $\langle a, b\rangle \in R$, então $\langle a, b\rangle \in$ FECHO-{transitiva}(R)

c.2) Se $\langle a, b\rangle \in$ FECHO-{transitiva}(R) e $\langle b, c\rangle \in$ FECHO-{transitiva}(R), então $\langle a, c\rangle \in$ FECHO-{transitiva}(R)

c.3) Os únicos elementos de FECHO-{transitiva}(R) são os construídos como acima.

A definição acima é um tipo especial de definição denominada de *definição indutiva* ou *definição recursiva* a qual é estudada em capítulo específico. Por enquanto, basta o entendimento intuitivo da construção.

Dois tipos de fecho de uma relação são especialmente importantes para Computação e Informática e possuem notação própria. Suponha uma endorrelação $R: A \rightarrow A$. Então:

a) *Fecho Transitivo* de R denotado por R^+, ou seja:

$$R^+ = \text{FECHO-}\{\text{transitiva}\}(R)$$

b) *Fecho Reflexivo e Transitivo* de R denotado por R^*, ou seja:

$$R^* = \text{FECHO-}\{\text{reflexiva, transitiva}\}(R)$$

EXEMPLO 6.7 - Fecho de uma Relação

Sejam $A = \{1, 2, 3, 4, 5\}$ um conjunto e $R: A \rightarrow A$ uma endorrelação tal que:

$$R = \{\langle 1, 2\rangle, \langle 1, 5\rangle, \langle 2, 3\rangle, \langle 3, 4\rangle\}$$

Então, alguns fechos de R é como segue (veja a Figura 6.6, da esquerda para a direita, respectivamente):

a) *Fecho Reflexivo.*

$$\text{FECHO-}\{\text{reflexiva}\}(R) = \{\langle 1, 1\rangle, \langle 1, 2\rangle, \langle 1, 5\rangle, \langle 2, 2\rangle, \langle 2, 3\rangle, \langle 3, 3\rangle, \langle 3, 4\rangle, \langle 4, 4\rangle, \langle 5, 5\rangle\}$$

b) *Fecho Simétrico.*

$$\text{FECHO-}\{\text{simétrica}\}(R) = \{\langle 1, 2\rangle, \langle 1, 5\rangle, \langle 2, 1\rangle, \langle 2, 3\rangle, \langle 3, 2\rangle, \langle 3, 4\rangle, \langle 4, 3\rangle, \langle 5, 1\rangle\}$$

c) *Fecho Transitivo.*

$$R^+ = \{\langle 1, 2\rangle, \langle 1, 3\rangle, \langle 1, 4\rangle, \langle 1, 5\rangle, \langle 2, 3\rangle, \langle 2, 4\rangle, \langle 3, 4\rangle\}$$

d) *Fecho Reflexivo e Transitivo.*

$$R^* = \{\langle 1, 1\rangle, \langle 1, 2\rangle, \langle 1, 3\rangle, \langle 1, 4\rangle, \langle 1, 5\rangle, \langle 2, 2\rangle, \langle 2, 3\rangle, \langle 2, 4\rangle, \langle 3, 3\rangle, \langle 3, 4\rangle, \langle 4, 4\rangle, \langle 5, 5\rangle\} \quad \square$$

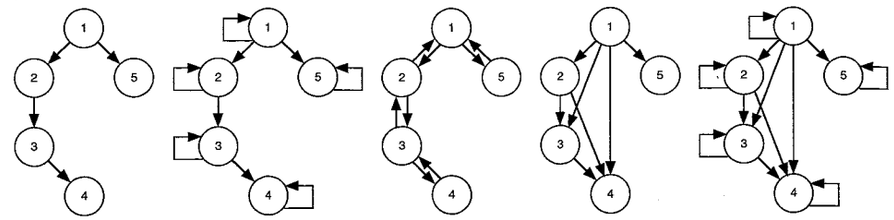


Figura 6.6 Grafos: relação e fechos reflexivo, simétrico, transitivo e transitivo/reflexivo

6.3 Ordenação

Um tipo especial e importante de relação é a *relação de ordem*, a qual reflete a noção intuitiva de ordem. Entre as relações já estudadas, são exemplos de relações de ordem:

- continência em conjuntos;
- implicação em proposições;
- menor ou igual (ou simplesmente menor) em algum conjunto numérico (como **R**, por exemplo).

Para o estudo das relações de ordem, é necessário introduzir a seguinte terminologia.

Definição 6.5 - Relação Conexa

Seja $R: A \rightarrow A$ uma endorrelação. Então R é uma *Relação Conexa*, se:

$$(\forall a \in A)(\forall b \in A)(a\,R\,b \vee b\,R\,a \vee a = b) \quad \square$$

EXEMPLO 6.8 - Relação Conexa

Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Então:

a) A endorrelação $\varnothing: B \rightarrow B$ é não-conexa;

b) A endorrelação $(C, <)$, dado que $<$ é definida por $\{\langle 0, 1\rangle, \langle 0, 2\rangle, \langle 1, 2\rangle\}$, é conexa;

c) A endorrelação $=: A \rightarrow A$ é conexa. $\square$

Para um entendimento intuitivo das propriedades de uma endorrelação de ordem considere, como exemplo, as filas de clientes para os diversos caixas de um banco. Então:

a) *Transitiva.* Uma importante propriedade da noção intuitiva de ordem. De fato, em uma fila, se João antecede José, e José antecede de Maria, então João antecede Maria;

b) *Anti-simétrica.* O princípio mais fundamental deste exemplo (e que melhor caracteriza qualquer ordem) é a propriedade anti-simétrica. De fato, considere a seguinte situação: a ordenado em relação à b e vice-versa (b ordenado em relação à a) só faz sentido se a for igual a b (no exemplo do banco, a antecede b e b antecede a só se a e b forem o mesmo cliente);

c) *Parcial/Conexa.* Nem todos os clientes estão relacionados entre si. De fato, em qualquer banco, sempre existe um caixa especial (e conseqüentemente, uma fila em separado) para idosos, gestantes e outros casos especiais. Entretanto, parcialidade não é uma propriedade fundamental para caracterizar formalmente a noção de ordem.

d) *Reflexiva/Irreflexiva.* Relações de ordem podem ser reflexivas (como, por exemplo, no caso da relação $\langle \mathbf{N}, \leq\rangle$) ou irreflexivas (como, por exemplo, no caso da relação $\langle \mathbf{Z}, <\rangle$). No exemplo das filas em um banco, o mais natural seria considerar como uma relação de ordem irreflexiva.

Entretanto, uma interpretação reflexiva (todo cliente antecede a si próprio na fila) faria sentido.

6.3.1 Relação de Ordem

Definição 6.6 - Relação de Ordem Parcial/Conexa, Ampla/Estrita
Seja R: A → A uma endorrelação. Então R é uma (veja a Figura 6.7):

a) *Relação de Ordem Parcial Ampla* ou simplesmente *Relação de Ordem Parcial*, se R é uma relação:
- reflexiva;
- anti-simétrica;
- transitiva;

b) *Relação de Ordem Parcial Estrita*, se R é uma relação:
- irreflexiva;
- anti-simétrica;
- transitiva;

c) *Relação de Ordem Conexa Ampla*, *Relação de Ordem Conexa* ou *Cadeia* se R é uma relação:
- de ordem parcial ampla;
- conexa;

d) *Relação de Ordem Conexa Estrita* ou *Cadeia Estrita* se R é uma relação:
- de ordem parcial estrita;
- conexa. ❑

	Ordem Parcial	Ordem Parcial Estrita	Cadeia	Cadeia Estrita
Reflexiva	✔		✔	
Irreflexiva		✔		✔
Anti-simétrica	✔	✔	✔	✔
Transitiva	✔	✔	✔	✔
Conexa			✔	✔

Figura 6.7 Propriedades dos diversos tipos de relações de ordem

Portanto, anti-simetria e transitividade são propriedades de qualquer tipo de relação de ordem, conforme ilustrado na Figura 6.7. Adicionalmente, toda relação de ordem conexa (ampla ou estrita, respectivamente) é uma relação de ordem parcial (ampla ou estrita, respectivamente), como ilustrado na Figura 6.8. A vice-versa nem sempre é verdadeira (por quê?).

Em uma relação de ordem (parcial/conexa, ampla/estrita) ⟨A, R⟩, o conjunto A é dito um *conjunto* (parcialmente/conexamente, amplamente/estritamente) *ordenado*. No caso em que a relação é de ordem parcial, também é denominado de *poset* (do inglês, *partially ordered set*).

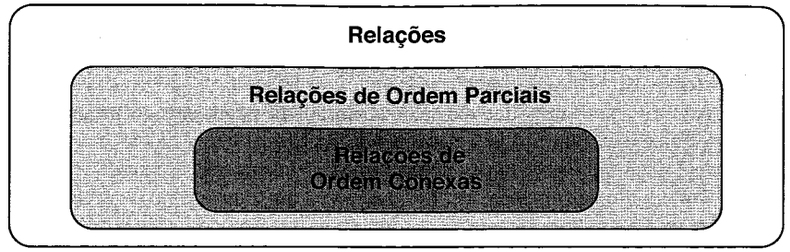


Figura 6.8 Continência entre os diversos tipos de relações de ordem

EXEMPLO 6.9 - Relação de Ordem Parcial/Conexa, Ampla/Estrita
Considere um conjunto não vazio A. Então, as seguintes relações são de:

a) Ordem parcial (ampla):
$\langle \mathbf{N}, \leq \rangle$
$\langle \mathbf{P}(A), \subseteq \rangle$
$\langle \mathbf{Q}, = \rangle$
implicação em proposições lógicas
$\{ \langle x, y \rangle \in \mathbf{N}^2 \mid x \text{ divide } y \text{ (resto zero)} \}$

b) Ordem parcial estrita:
$\langle \mathbf{N}, < \rangle$
$\langle \mathbf{P}(A), \subset \rangle$

c) Ordem conexa (ampla):
$\langle \mathbf{N}, \leq \rangle$

d) Ordem conexa estrita:
$\langle \mathbf{N}, < \rangle$ ❑

EXEMPLO 6.10 - Ordem Lexicográfica
Ordem lexicográfica é um importante exemplo de relação de ordem conexa para Computação e Informática. Lembre-se de que, por exemplo, para um dado alfabeto $\Sigma = \{a, b\}$, o conjunto Σ^* (todas as palavras sobre Σ), é como segue:

$$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$$

Observe que as palavras, no conjunto Σ^*, foram listadas em ordem lexicográfica, ou seja, por tamanho de palavra (número de símbolos) e, para palavras do mesmo tamanho, por ordem "alfabética" (supondo que a é menor do que b).

Assim, qualquer alfabeto ordenado Σ induz o conjunto ordenado (lexicograficamente) Σ^*. ❑

6.3.2 Classificação de Dados

Uma importante questão para Computação e Informática é a ordenação de um conjunto de dados. De fato, trata-se de uma importante área de pesquisa, usualmente denominada de *classificação de dados* (em inglês, *sort*). Embora não seja difícil construir um algoritmo de classificação, o assunto é delicado pois, na medida em que o número de dados aumenta, o tempo (de processamento) e o espaço (memória de armazenamento) podem se tornar críticos. O estudo

do tempo/espaço consumidos por um algoritmo também é uma importante área de pesquisa denominada de *complexidade de algoritmos*.

EXEMPLO 6.11 - Algoritmo de Classificação

Já foi comentado diversas vezes que a grande maioria das linguagens de programação não possuem boas facilidades para manipular conjuntos. Assim, a ordenação de um conjunto de dados usualmente é realizada em variáveis do tipo *arranjo*. Uma variável arranjo é uma seqüência com número fixo e finito de componentes. Por sua vez, cada componente da seqüência é uma variável, sempre do mesmo tipo. Ou seja, uma variável do tipo arranjo é uma seqüência finita de variáveis, todas do mesmo tipo. Por exemplo, os seguintes trechos de programa em Pascal:

```
vetor = array[1..30] of integer
```

```
dados = array[1..10] of char
```

definem duas variáveis do tipo arranjo, como segue:

- `vetor` é uma seqüência de 30 componentes do tipo inteiro
- `dados` é uma seqüência de 10 componentes do tipo caractere.

Cada componente pode ser diretamente acessada, especificando-se o nome da variável arranjo seguido do índice (posição relativa do componente) entre colchetes. Por exemplo (suponha `i` uma variável do tipo inteiro):

```
vetor[10] := 33
```

```
if dados[i] = 'a' then ...
```

definem dois comandos como segue, respectivamente:

- atribui o valor 33 à décima componente da variável arranjo `vetor`;
- se a i-ésima componente da variável arranjo `dados` for "a", então...

Suponha que se deseja ordenar (relação "menor ou igual") um conjunto de 10 valores do tipo caractere os quais estão armazenados em uma variável arranjo `dados`. Ou seja, deseja-se obter:

`dados[1] ≤ dados[2] ≤ dados[3] ≤ ... ≤ dados[10]`

Uma solução (trecho de programa Pascal) é apresentada na Figura 6.9 (suponha que `trocou` e `aux` são variáveis do tipo `boolean` e `char`, respectivamente). O algoritmo em questão é conhecido como *bubblesort*. O próprio nome do algoritmo (borbulha, tradução do termo em inglês *bubble*) sugere o seu funcionamento: os dados mais "leves" gradativamente sobem, ficando acima dos mais "pesados". Observe que:

- a idéia geral do algoritmo consiste em verificar se, entre duas componentes vizinhas, a ordem desejada é respeitada. Se não for, troca as componentes vizinhas. Este ciclo é repetido enquanto houver alguma troca;
- o ciclo de repetição é definido usando um comando `while-do` (já apresentado);
- para garantir que o ciclo é realizado pelo menos uma vez (no melhor caso, os dados já estão ordenados), a variável `trocou` é inicializada com o valor `true`;
- dentro do ciclo, inicialmente é atribuído o valor `false` para a variável `trocou`. Entretanto, se no ciclo ocorrer alguma troca, `trocou` recebe o valor `true`;
- o comando `for-do` tem a seguinte semântica: o comando após a palavra `do` é executado repetidamente para `i` variando de 1 até 9. A razão para a variável `i` assumir valores até 9 (dado que são 10 componentes) é que o algoritmo acessa a componente `i+1`;

- caso a ordem de duas componentes vizinhas esteja invertida (`dados[i] > dados[i+1]`), é realizada a troca de valores;
- é usada uma variável auxiliar `aux` para realizar a troca. Intuitivamente, o seguinte trecho de programa, sem variável auxiliar, seria suficiente:

```
dados[i] := dados[i+1];
dados[i+1] := dados[i];
```

- entretanto, nesse caso, ambas as componentes ficariam com o valor da componente `dados[i+1]` (por quê?).

Uma possível execução do algoritmo é apresentada na tabela ilustrada na Figura 6.10. Na primeira linha, é apresentada a distribuição inicial dos valores nas 10 componentes do arranjo. Nas linhas subseqüentes, como ficam as componentes do arranjo a cada interação do comando `while-do`?

O algoritmo proposto, embora eficiente em termos de espaço (não necessita de memória além do arranjo que já armazena os dados a serem classificados), não é eficiente em termos do tempo de processamento, para grandes volumes de dados. ❑

```
trocou := true;
while trocou
do    begin
      trocou := false;
      for   i := 1 to 9
      do    if    dados[i] > dados[i+1]
            then  begin
                  aux := dados[i];
                  dados[i] := dados[i+1];
                  dados[i+1] := aux;
                  trocou := true
                  end
      end
```

Figura 6.9 Classificação: trecho de programa em Pascal

Inicial	c	a	d	b	a	b	d	f	e	f
Interação 1	a	c	b	a	b	d	d	e	f	f
Interação 2	a	b	a	b	c	d	d	e	f	f
Interação 3	a	a	b	b	c	d	d	e	f	f
Interação 4	a	a	b	b	c	d	d	e	f	f

Figura 6.10 Classificação: exemplo de execução do trecho de programa Pascal

6.3.3 Diagrama de Hasse

Qualquer relação de ordem pode ser representada na forma de um grafo, como qualquer outra endorrelação, como ilustrado na Figura 6.6 (quais relações representadas são de ordem? Nesse caso, de que tipo?). A principal característica de uma relação de ordem como grafo é que jamais ocorrerá um ciclo (por quê?), excetuando-se, eventualmente, no caso de *endoarcos* (ou *endoarestas*), ou seja, arcos com origem e destino em um mesmo nodo.

Entretanto, quando se trata de uma relação de ordem, existe uma certa "poluição visual" ocasionada pela propriedade transitiva. O mesmo ocorre com as endoarestas de uma relação de ordem ampla, ocasionada pela propriedade reflexiva. Assim, o usual é representar a relação de ordem, omitindo-se as arestas que podem ser deduzidas pelas propriedades transitiva e reflexiva. Como ilustração, considere a Figura 6.6, da esquerda para a direita:

- a relação de ordem estrita representada pelo *quarto* grafo é usualmente representada, de forma simplificada, pelo *primeiro* grafo;
- a relação de ordem (ampla) representada pelo *quinto* grafo também é usualmente representada, de forma simplificada, pelo *primeiro* grafo.

Claramente, nos dois casos acima, as arestas omitidas podem ser recuperadas pelo fecho transitivo (respectivamente, fecho transitivo e reflexivo). Esse tipo de representação de relações de ordem é denominado *Diagrama de Hasse* (Helmut Hasse (1898-1979), matemático alemão). Em diagramas de Hasse, também é usual representar os nodos por pontos (ou pequenos círculos) ou simplesmente pelo elemento do conjunto como origem ou destino das arestas.

EXEMPLO 6.12 - Relação como Grafo × Diagrama de Hasse

Considere o conjunto parcialmente ordenado $\langle \{1,2,3\}, \leq \rangle$ sendo que:

$$\leq = \{\langle 1,1\rangle, \langle 2,2\rangle, \langle 3,3\rangle, \langle 1,2\rangle, \langle 2,3\rangle, \langle 1,3\rangle\}$$

Assim, a relação de ordem é representada na Figura 6.11, como grafo (esquerda) e como diagrama de Hasse (direita).

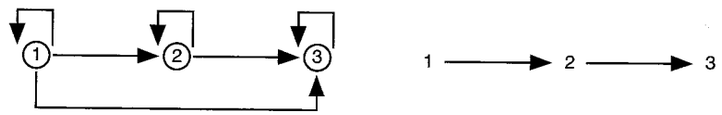


Figura 6.11 Relação como Grafo × Diagrama de Hasse

Observação 6.7 - Representações Alternativas para Diagrama de Hasse

Uma representação alternativa muito comum para um diagrama de Hasse é usar arestas não-orientadas. Neste caso, os elementos são distribuídos no diagrama do menor para o maior, de baixo para cima, respectivamente, como ilustrado na Figura 6.12 para a relação de ordem $\langle \{1,2,3\}, \leq \rangle$ (compare com a representação da mesma relação mostrada na Figura 6.11). ❑

Figura 6.12 Representação alternativa usual para Diagrama de Hasse

6.3.4 Conjuntos Ordenados e Semântica de Sistemas Concorrentes

Conjuntos ordenados (ou estruturas baseadas em conjuntos ordenados) são usados com freqüência em Computação e Informática. Um importante exemplo é na semântica para *sistemas*

concorrentes. De fato, conjuntos ordenados fornecem uma visão clara e simples de concorrência, no sentido verdadeiro da palavra, usualmente denominada *concorrência verdadeira*.

Inicialmente, é importante distinguir sintaxe de semântica:

a) *Sintaxe* trata das propriedades livres de uma linguagem como, por exemplo, a verificação gramatical de programas;
b) *Semântica* objetiva dar uma interpretação como, por exemplo, um significado ou um valor a um programa.

Portanto:

- a sintaxe preocupa-se com a forma, manipulando símbolos;
- a semântica preocupa-se em dar um significado aos símbolos sintaticamente válidos como, por exemplo, "estes símbolos representam os valores inteiros".

Questões sintáticas e semânticas são usualmente desenvolvidas em disciplinas como *Linguagens Formais* e *Semântica Formal*, respectivamente. Observe que a disciplina de *Compiladores* integra ambas as questões.

Historicamente, no estudo do entendimento das linguagens de programação, o problema sintático foi reconhecido antes do problema semântico e foi o primeiro a receber um tratamento adequado. Adicionalmente, os problemas sintáticos são de tratamento mais simples que os semânticos. Como conseqüência, foi dada uma grande ênfase à sintaxe, a ponto de levar à idéia de que as questões das linguagens de programação resumiam-se às questões da sintaxe. Atualmente, a teoria da sintaxe possui construções matemáticas bem definidas e universalmente reconhecidas como, por exemplo, as *Gramáticas de Chomsky*. O mesmo não pode ser afirmado para a teoria da semântica. O problema não é somente a sua maior complexidade. A formalização de uma questão semântica pode, freqüentemente, possuir um tratamento matemático extremamente complexo, dificultando o seu entendimento e a sua aplicação na prática.

Assim, qualquer construção matemática capaz de dar semântica de forma simples e expressiva é extremamente importante para a Computação e Informática. Um bom exemplo são as relações de ordem como semântica dos sistemas concorrentes.

A semântica de sistemas concorrentes usando conjuntos ordenados é apresentada na forma de exemplos. Usualmente são usadas relações de ordem parciais as quais podem ser amplas ou estritas. No que segue, é adotada a estrita (irreflexiva).

EXEMPLO 6.13 - Conjuntos Parcialmente Ordenados × Concorrência

Considere o seguinte trecho de programa seqüencial, baseado em linguagens de programação do tipo Pascal, na qual o símbolo `;` representa uma relação de dependência causal dos comandos `c1`, `c2` e `c3`:

```
c1; c2; c3
```

Uma semântica de tal programa pode ser dada pelo seguinte conjunto parcialmente ordenado:

$$\langle \{c1, c2, c3\}, \leq_c \rangle \quad \text{onde} \quad c1 \leq_c c2 \text{ e } c2 \leq_c c3$$

e, portanto, $c1 \leq_c c3$. Mais precisamente:

$$\leq_c = \{(c1, c2), (c2, c3), (c1, c3)\}$$

De forma análoga, considere os seguintes trechos de programas:

```
p1; p2
```

```
q1; q2; q3
```

e as seguintes correspondentes semânticas:

$$\langle \{p1, p2\}, \leq_p \rangle \quad \text{onde} \quad p1 \leq_p p2$$
$$\langle \{q1, q2, q3\}, \leq_q \rangle \quad \text{onde} \quad q1 \leq_q q2 \text{ e } q2 \leq_q q3$$

e, portanto, $q1 \leq_q q3$. Suponha que os três trechos de programa são concorrentes sem qualquer sincronização (são *independentes*). Então a semântica de tal programa concorrente seria simplesmente o seguinte conjunto ordenado (induzido pela operação de união disjunta de conjuntos):

$$\langle \{c1, c2, c3\} + \{p1, p2\} + \{q1, q2, q3\}, \leq_c + \leq_p + \leq_q \rangle$$

representado na forma de diagrama de Hasse na Figura 6.13 (esquerda). Uma característica importante é que, segundo essa abordagem, todas as componentes de um conjunto ordenado são independentes (concorrentes), exceto quando for especificado o contrário (ou seja, quando for definido um par da relação de ordem determinando uma restrição de seqüencialidade).

Adicionalmente, suponha que a ocorrência de p2 e c3 depende de c2 e q3, respectivamente, como representado da Figura 6.13 (direita). Para expressar tal dependência causal (sincronização) dos três trechos de programa, é suficiente incluir os seguintes pares na união disjunta:

$$c2 \leq p2 \quad \text{e} \quad q3 \leq c3$$

ou seja:

$$\langle \{c1, c2, c3\} + \{p1, p2\} + \{q1, q2, q3\}, \leq_c + \leq_p + \leq_q + \{(c2, p2), (q3, c3)\} \rangle \quad \square$$

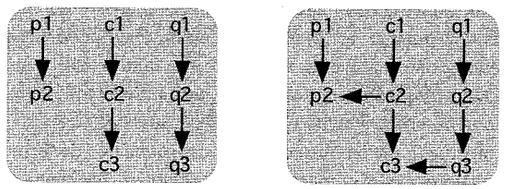


Figura 6.13 Conjuntos parcialmente ordenados como modelo para expressar concorrência

No exemplo acima, é interessante observar que:

- a operação *união disjunta* de conjuntos ordenados pode ser interpretada como *composição paralela* de sistemas;
- a inclusão de pares na relação de ordem parcial pode determinar *sincronizações*.

Ou seja, operações simples e de fácil entendimento constituem operadores poderosos para especificar *sistemas concorrentes e comunicantes* em termos de suas partes componentes (sistemas mais "elementares").

Observação 6.8 - Estrutura de Eventos

Um dos modelos para concorrência mais conhecidos baseados em conjuntos ordenados é a *Estrutura de Eventos*, a qual é constituída, basicamente, por um conjunto ordenado (para especificar as noções de seqüencialidade e de concorrência) juntamente com uma relação de conflito (para especificar a noção de *não-determinismo* ou *escolha*). O conceito de não-determinismo é introduzido ao longo do livro. $\square$

Um exercício interessante para verificar a expressividade dos conjuntos parcialmente ordenados é o seguinte: caso você conheça alguma linguagem de programação concorrente, faça um esboço de um programa concorrente para o caso exemplificado acima e compare as

especificações. De fato, comparativamente com muitas das linguagens usualmente adotadas, os conjuntos parcialmente ordenados fornecem soluções mais simples e claras.

6.4 Equivalência e Partição

Assim como a relação de ordem, a *relação de equivalência* também é importante para Computação e Informática e reflete uma noção de igualdade semântica, no sentido em que entidades com formas diferentes (ou seja, sintaticamente diferentes) podem ser equivalentes ("igualadas"). Uma relação de equivalência já introduzida que ilustra essa noção de igualdade semântica é a relação lógica $\Leftrightarrow$ (lembre-se de que duas proposições lógicas diferentes podem ser equivalentes). Outros exemplos podem ser facilmente obtidos no cotidiano. Como ilustração, considere um conjunto de pessoas. Então, são relações de equivalência:

- mesma idade;
- mesma altura;
- mesmo sexo.

Considerando que a equivalência reflete uma noção semântica de igualdade, então as seguintes propriedades são intuitivas e caracterizam qualquer relação de equivalência:

a) *Reflexiva*. De fato, qualquer elemento é sempre "igual" a si mesmo;
b) *Transitiva*. Uma importante propriedade da noção intuitiva de "igualdade";
c) *Simétrica*. Provavelmente, a noção que mais caracteriza a "igualdade" (e diferencia da noção de ordem).

Um importante resultado apresentado adiante é que cada relação de equivalência $R: A \rightarrow A$ induz uma única *partição do conjunto* A em subconjuntos disjuntos e não-vazios denominados *classes de equivalência*. No exemplo do conjunto de pessoas, tem-se as seguintes classes de equivalência, considerando a relação "mesmo sexo":

- classe de equivalência das pessoas do sexo feminino;
- classe de equivalência das pessoas do sexo masculino.

Portanto, os conceitos de relação de equivalência e de partição de conjuntos estão diretamente relacionados.

Definição 6.9 - Relação de Equivalência

Seja $R: A \rightarrow A$ uma endorrelação. Então R é uma *Relação de Equivalência* se for reflexiva, simétrica e transitiva. $\square$

Portanto, como qualquer relação de equivalência, $R: A \rightarrow A$ é reflexiva, o domínio de definição e o conjunto imagem coincidem com A.

Definição 6.10 - Partição de um Conjunto

Seja A um conjunto. Uma *partição do conjunto* A é um conjunto de subconjuntos não-vazios e mutuamente disjuntos de A, denominados de *classes de equivalência* ou *blocos da partição*, tal que a união de todos os blocos resulta em A. $\square$

Leia atentamente a definição acima. Observe que A pode ser vazio. Então, pela definição, quais são os blocos da partição do vazio?

Suponha que, para dado conjunto A, $\{A_1, A_2, \ldots, A_n\}$ é uma partição de A. Então, é usual denotar uma classe de equivalência por um elemento representativo desta, entre colchetes. Assim, se $a_1 \in A_1, a_2 \in A_2, \ldots, a_n \in A_n$, tem-se que:

$$[a_1] = A_1, [a_2] = A_2, \ldots, [a_n] = A_n$$

Observe que a notação de uma classe de equivalência por um elemento representativo é usual na vida cotidiana. Por exemplo, no Código Nacional de Trânsito, entre os sinais de advertência estão duas placas com os seguintes símbolos:

- uma vaca, para representar genericamente "cuidado animais";
- um alce, para representar genericamente "cuidado animais selvagens".

Ou seja, a vaca e o alce são animais (elementos) representativos das classes animais e animais selvagens, respectivamente.

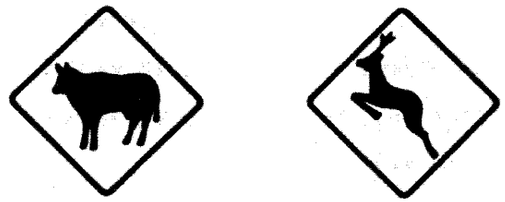


Figura 6.14 Código Nacional de Trânsito: sinais de advertência

EXEMPLO 6.14 - Relação de Equivalência

Considere um conjunto A. Então, são relações de equivalência (procure induzir qual seria a correspondente partição em cada caso):

$\langle A, = \rangle$
$\langle \mathbf{P}(A), = \rangle$
$\varnothing : \varnothing \to \varnothing$
$A^2 : A \to A$ ❑

EXEMPLO 6.15 - Relação de Equivalência e Partição

Considere a seguinte relação:

$$R = \{\langle a, b\rangle \in \mathbf{N}^2 \mid a \text{ MOD } 2 = b \text{ MOD } 2\}$$

onde MOD é a operação que resulta no resto da divisão inteira. É fácil verificar que R é uma relação de equivalência.

Intuitivamente, a relação R induz uma partição do conjunto **N**, como segue:

[0], a classe de equivalência dos números pares (resto zero);
[1], a classe de equivalência dos números ímpares (resto um). ❑

O seguinte teorema mostra como construir uma partição a partir de uma relação de equivalência. A prova é especialmente interessante pois, embora simples, usa três técnicas de demonstração: direta, contraposição e absurdo.

Teorema 6.11 - Relação de Equivalência ⇒ Partição

Seja $R: A \to A$ uma relação de equivalência. Então, R induz uma partição do conjunto A.

Prova:

Suponha que $R: A \to A$ é uma relação de equivalência. Para qualquer $a \in A$, seja:

$$[a]_R = \{b \in A \mid a R b\}$$

Então, o seguinte conjunto é uma partição de A:

$$\{[a]_R \mid a \in A\}$$

Para provar que, de fato, esse conjunto é uma partição de A, é necessário provar que:

- cada classe de equivalência é não-vazia;
- quaisquer duas classes de equivalência distintas são disjuntas;
- a união de todas as classes de equivalência resulta no conjunto A.

Cada classe de equivalência é não-vazia (prova *direta*). Suponha $a \in A$. Então:

$a \in A \Rightarrow$	reflexividade de R
$a R a \Rightarrow$	definição de $[a]_R$
$a \in [a]_R$	

Logo, cada classe de equivalência é não-vazia.

Quaisquer duas classes de equivalência distintas são disjuntas. Para atingir esse resultado, inicialmente, é necessário provar um resultado sobre classes de equivalência distintas:

a) Se $[a]_R \neq [b]_R$, então $\neg(a R b)$ (prova por *contraposição*). Suponha que $a R b$. Então, a prova de que $[a]_R = [b]_R$ é dividida em dois casos (duas continências), como segue:

Caso 1. $[b]_R \subseteq [a]_R$. Suponha $c \in [b]_R$:

$c \in [b]_R \Rightarrow$	definição de $[b]_R$
$b R c \Rightarrow$	transitividade de R suposto que $a R b$
$a R c \Rightarrow$	definição de $[a]_R$
$c \in [a]_R \Rightarrow$	definição de subconjunto
$[b]_R \subseteq [a]_R$	

Caso 2. $[a]_R \subseteq [b]_R$. Suponha $c \in [a]_R$:

$c \in [a]_R \Rightarrow$	definição de $[a]_R$
$a R c \Rightarrow$	simetria de R
$c R a \Rightarrow$	transitividade de R suposto que $a R b$
$c R b \Rightarrow$	simetria de R
$b R c \Rightarrow$	definição de $[b]_R$
$c \in [b]_R \Rightarrow$	definição de subconjunto
$[a]_R \subseteq [b]_R$	

Logo, se $[a]_R \neq [b]_R$, então $\neg(a R b)$

b) Se $[a]_R \neq [b]_R$, então $[a]_R \cap [b]_R = \varnothing$ (*por absurdo*). Suponha que $[a]_R \neq [b]_R$ e $[a]_R \cap [b]_R \neq \varnothing$. Então:

$[a]_R \neq [b]_R \wedge [a]_R \cap [b]_R \neq \varnothing \Rightarrow$	item a)
$\neg(a R b) \wedge [a]_R \cap [b]_R \neq \varnothing \Rightarrow$	definição de intersecção
$\neg(a R b) \wedge (\exists c \in A)(c \in [a]_R \wedge c \in [b]_R) \Rightarrow$	definição de $[a]_R$ e de $[b]_R$
$\neg(a R b) \wedge a R c \wedge b R c \Rightarrow$	simetria de R
$\neg(a R b) \wedge a R c \wedge c R b \Rightarrow$	transitividade de R
$\neg(a R b) \wedge a R b$, o que é um *absurdo*!	

Logo, quaisquer duas classes de equivalência distintas são disjuntas.

União de todas as classes de equivalência resulta no conjunto A (prova *direta*). A prova é dividida em dois casos (duas continências), como segue:

Caso 1. A está contido na união. Suponha $a \in A$:

$a \in A \Rightarrow$ classe de equivalência é não-vazia
$a \in [a]_R \Rightarrow$ definição de união
a pertence à união de todas as classes de equivalência

Caso 2. União está contida em A. Suponha que a pertence à união de todas as classes:

a pertence à união de todas as classes $\Rightarrow$ definição de união
$(\exists b \in A)(a \in [b]_R) \Rightarrow$ definição de classe
$b\,R\,a \Rightarrow$ suposto que $R: A \rightarrow A$
$a \in A$

Logo, a união de todas as classes de equivalência resulta no conjunto A.

Portanto, R induz uma partição do conjunto A. ❑

Portanto, para construir a partição induzida pela relação basta agrupar os elementos que estão relacionados entre si como uma classe de equivalência.

EXEMPLO 6.16 - Relação de Equivalência ⇒ Partição

Considere a seguinte relação apresentada no EXEMPLO 6.15 - Relação de Equivalência e Partição:

$$R = \{\langle a, b\rangle \in \mathbf{N}^2 \mid a \text{ MOD } 2 = b \text{ MOD } 2\}$$

Claramente, a R b se e somente se os naturais a e b, quando dividido por 2, ou têm ambos resto zero ou ambos resto um, ou seja, são ambos pares ou ambos ímpares. Portanto, como já introduzido intuitivamente, R induz a seguinte partição em **N**:

[0], a classe de equivalência dos números pares (resto zero);
[1], a classe de equivalência dos números ímpares (resto um). ❑

Os seguintes teoremas não serão provados.

Teorema 6.12 - Partição Induzida por uma Relação de Equivalência é Única

Seja $R: A \rightarrow A$ uma relação de equivalência. Então, a partição do conjunto A induzida por R é *única*. ❑

Teorema 6.13 - Partição ⇒ Relação de Equivalência

Seja A um conjunto. Então, qualquer partição do conjunto A induz uma relação de equivalência $R: A \rightarrow A$. ❑

Definição 6.14 - Conjunto Quociente

Seja A um conjunto e $R: A \rightarrow A$ uma endorrelação de equivalência. O *Conjunto Quociente* denotado como segue:

$$A/R$$

é a partição de A induzida pela relação de equivalência R como visto no Teorema 6.11 - Relação de Equivalência ⇒ Partição, ou seja:

$$A/R = \{[a]_R \mid a \in A\}$$ ❑

EXEMPLO 6.17 - Conjunto Quociente: Conjunto dos Números Racionais

Suponha que $\mathbf{N}_+$ denota o conjunto dos números naturais positivos, ou seja:

$$\mathbf{N}_+ = \mathbf{N} - \{0\}$$

e que **F** denota o seguinte conjunto, freqüentemente denominado de conjunto das frações (por que excluir o natural zero da segunda componente?):

$$\mathbf{F} = \mathbf{Z} \times \mathbf{N}_+$$

Considere a seguinte relação de equivalência:

$$R = \{\langle\langle a, b\rangle, \langle c, d\rangle\rangle \in \mathbf{F}^2 \mid a/b = c/d\}$$

Portanto, o conjunto quociente $\mathbf{F}/R$ é o conjunto dos números racionais, ou seja:

$$\mathbf{Q} = \mathbf{F}/R$$

Assim, cada número racional é, de fato, uma classe de equivalência de frações como, por exemplo (neste contexto, é usual denotar um par $\langle a, b\rangle$ na forma a/b):

$[0]_R = \{0/1, 0/2, 0/3, \ldots\}$
$[1/2]_R = \{1/2, 2/4, 3/6, \ldots\}$
$[5/4]_R = \{5/4, 10/8, 15/12, \ldots\}$ ❑

6.5 Exercícios

Exercício 6.1 Seja $A = \{a, b\}$. Determine todas as endorrelações em A e verifique quais são:

a) Reflexivas;
b) Irreflexivas;
c) Simétricas;
d) Anti-simétricas;
e) Transitivas;
f) Conexas.

Exercício 6.2 Seja $A = \{1, 2, 3\}$. Para cada uma das seguintes endorrelações em A, determine se é:

- reflexiva;
- irreflexiva;
- simétrica;
- anti-simétrica;
- transitiva;
- conexa.

a) $R_1 = \{\langle 1, 2\rangle, \langle 1, 1\rangle, \langle 2, 2\rangle, \langle 2, 1\rangle, \langle 3, 3\rangle\}$
b) $R_2 = \{\langle 1, 1\rangle, \langle 2, 2\rangle, \langle 3, 3\rangle, \langle 1, 2\rangle, \langle 2, 3\rangle\}$
c) $R_3 = \{\langle 1, 1\rangle, \langle 2, 2\rangle, \langle 1, 2\rangle, \langle 2, 3\rangle, \langle 3, 1\rangle\}$
d) $R_4 = A \times A$
e) $R_5 = \varnothing$

Exercício 6.3 Seja $A = \{a, b, c, d\}$. Defina endorrelações em A tais que:

a) R_1: só tem a propriedade reflexiva;
b) R_2: só tem a propriedade simétrica;
c) R_3: só tem a propriedade transitiva;

d) R_4: só tem a propriedade anti-simétrica;
e) R_5: reflexiva e transitiva, mas não-simétrica;
f) R_6: reflexiva e simétrica, mas não-transitiva;
g) R_7: simétrica e transitiva, mas não-reflexiva.

Exercício 6.4 Defina uma endorrelação que é simultaneamente reflexiva *e* irreflexiva e determine os correspondentes grafo e matriz.

Exercício 6.5 Como seria a matriz e o grafo de uma endorrelação que *não* é reflexiva *nem* irreflexiva?

Exercício 6.6 Exemplifique cada um dos casos abaixo:

a) Relação que *não* é simétrica *nem* anti-simétrica;
b) Relação que é simultaneamente simétrica *e* anti-simétrica.

Exercício 6.7 Para cada um dos fechos ilustrados na Figura 6.6, faça a correspondente matriz.

Exercício 6.8 Seja $A = \{1, 2, 3\}$. Para cada uma das endorrelações em A abaixo, determine o seguinte:

- fecho reflexivo e transitivo;
- fecho simétrico;

e represente o resultado como:

- grafo;
- matriz.

a) $R_0 = \{\langle 1,1\rangle, \langle 1,2\rangle, \langle 2,1\rangle\}$
b) $R_1 = \{\langle 1,1\rangle, \langle 1,2\rangle, \langle 2,1\rangle, \langle 2,2\rangle, \langle 3,3\rangle\}$
c) $R_2 = \{\langle 1,1\rangle, \langle 1,2\rangle, \langle 2,2\rangle, \langle 2,3\rangle, \langle 3,3\rangle\}$
d) $R_3 = \{\langle 1,1\rangle, \langle 1,2\rangle, \langle 2,2\rangle, \langle 2,3\rangle, \langle 3,1\rangle\}$
e) $R_4 = A \times A$
f) $R_5 = \varnothing$

Exercício 6.9 Em que condições uma endorrelação R é igual ao seu fecho, ou seja, $R = \text{FECHO-P}(R)$?

Exercício 6.10 Faz sentido pensar nos seguintes fechos? Justifique a sua resposta:

a) Fecho irreflexivo;
b) Fecho anti-simétrico.

Exercício 6.11 Suponha que são conhecidos todos os trechos parciais que podem ser percorridos por um carteiro (exemplo: da casa A para a casa B). Usando a noção de grafo como uma relação e o conceito de fecho, como se pode representar todos os caminhos possíveis que o carteiro pode fazer?

Exercício 6.12 Sejam $A = \{2, 3, 4, 5\}$ e $B = \{3, 4, 5, 6, 10\}$. Para cada uma das seguintes relações, verifique se são conexas:

a) $R_1 = \{\langle x, y\rangle \in B \times B \mid x \text{ é divisível por } y\}$
b) $R_2 = \{\langle x, y\rangle \in A \times A \mid x * y = 12\}$
c) $R_3 = \{\langle x, y\rangle \in A \times A \mid x = y + 1\}$
d) $R_4 = \{\langle x, y\rangle \in B \times B \mid x \leq y\}$

Exercício 6.13 As *relações de ordem* são fechadas para as seguintes operações sobre conjuntos (ou seja, a operação de duas relações de ordem resulta em uma relação de ordem)? Justifique a sua resposta:

a) União;
b) Intersecção;
c) Complemento;
d) Diferença;
e) Produto Cartesiano.

Exercício 6.14 Foi visto que toda relação de ordem conexa (ampla ou estrita, respectivamente) é uma relação de ordem parcial (ampla ou estrita, respectivamente), como ilustrado na Figura 6.8. Justifique por que a vice-versa nem sempre é verdadeira.

Exercício 6.15 Mostre que as seguintes relações de ordem não são conexas:

a) Para um dado conjunto A, $\langle \mathbf{P}(A), \subseteq\rangle$;
b) Relação implicação em proposições.

Exercício 6.16 Por que, em uma relação de ordem vista como um grafo, jamais ocorrerá um ciclo (um caminho partindo e chegando em um mesmo nodo), excetuando-se, eventualmente, no caso de *endoarcos* (ou *endoarestas*) ou seja, arcos com origem e destino em um mesmo nodo?

Exercício 6.17 Suponha que a relação de ordem ilustrada na forma de um diagrama de Hasse na Figura 6.15 representa um sistema concorrente. Então:

a) Quais os elementos independentes?
b) Para cada elemento não-independente, explicite os elementos dos quais ele depende.

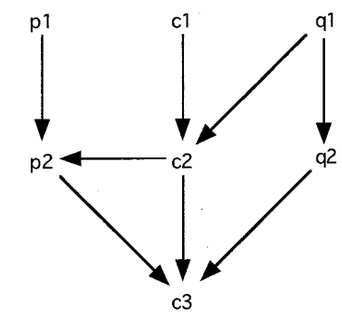


Figura 6.15 Conjuntos parcialmente ordenados como um sistema concorrente

Exercício 6.18 Um exercício interessante para verificar a expressividade dos conjuntos parcialmente ordenados é o seguinte: caso você conheça alguma linguagem de programação concorrente, faça um esboço de um programa concorrente para o caso exemplificado na Figura 6.15 e compare as especificações. De fato, comparativamente com muitas das linguagens usualmente adotadas, os conjuntos parcialmente ordenados fornecem soluções mais simples e claras.

Exercício 6.19 Seja $A = \{a, b, c\}$. Determine quais das seguintes endorrelações em A são relações de equivalência (justifique a sua resposta):

a) $R_1 = \{\langle a, a\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle b, b\rangle, \langle c, c\rangle\}$

b) $R_2 = \{\langle a, a\rangle, \langle a, b\rangle, \langle b, a\rangle, \langle b, b\rangle, \langle b, c\rangle\}$
c) $R_3 = \{\langle a, a\rangle, \langle b, b\rangle, \langle b, c\rangle, \langle c, b\rangle, \langle a, c\rangle, \langle c, a\rangle\}$
d) $R_4 = A \times A$
e) $R_5 = \varnothing$

Exercício 6.20 Em que condições a relação $\varnothing: A \to A$ é uma relação de equivalência?

Exercício 6.21 Seja R uma endorrelação em $\mathbf{N}^2$ definida por:

$$\langle a, b\rangle R \langle c, d\rangle \Leftrightarrow a + b = c + d$$

Demonstre que R é uma relação de equivalência.

Exercício 6.22 Seja R uma endorrelação em $\mathbf{N}^2$ definida por:

$$\langle a, b\rangle R \langle c, d\rangle \Leftrightarrow a * d = b * c$$

Demonstre que R não é uma relação de equivalência.

Exercício 6.23 As *relações de equivalência* são fechadas para as seguintes operações sobre conjuntos (ou seja, a operação de duas relações de equivalência resulta em uma relação de equivalência)? Justifique a sua resposta:

a) União;
b) Intersecção;
c) Complemento;
d) Diferença;
e) Produto Cartesiano.

Exercício 6.24 Seja $A = \{1, 2, 3\}$. Para cada uma das seguintes relações de equivalência, apresente a partição induzida (ou seja, o correspondente conjunto quociente):

a) $\langle A, =\rangle$
b) $\langle \mathbf{P}(A), =\rangle$
c) $\varnothing: \varnothing \to \varnothing$
d) $A^2: A \to A$

Exercício 6.25 Sejam $A = \{x \mid 0 \le x \le 10\}$ e R uma endorrelação em A definida por:

$$x R y \Leftrightarrow (\exists k \in \mathbf{Z})\ (x - y = 4k)$$

Qual o conjunto quociente A/R?

Exercício 6.26 Complemente a implementação do sistema proposto no Capítulo 4 - Relações (o qual permite definir relações e tratar construções correlatas) de tal forma que também seja capaz de:

- verificar as propriedades de uma endorrelação;
- para um conjunto de propriedades selecionadas, calcular o fecho de uma endorrelação.

7 Cardinalidade de Conjuntos

Entende-se por *cardinalidade* de um conjunto uma medida de seu tamanho. Até o momento, a cardinalidade de conjuntos vem sendo tratada de maneira informal ou semi-formal. Por exemplo, a seguinte expressão foi usada com alguma freqüência:

número de elementos de um conjunto

Também já foi afirmado que:

um conjunto pode possuir um número finito ou infinito de elementos

Nesse contexto, foi afirmado que um conjunto é:

a) *Conjunto finito* se pode ser denotado por extensão, ou seja, listando exaustivamente todos os seus elementos;
b) *Conjunto infinito*, caso contrário.

Assim, em um estudo mais formal sobre a cardinalidade de conjuntos, as seguintes perguntas surgem naturalmente:

- como definir formalmente a cardinalidade de um conjunto?
- quando dois conjuntos possuem o mesmo cardinal?
- o que é um cardinal infinito?
- existe mais de um, ou seja, existem diferentes cardinais infinitos?
- nesse caso, existe uma ordem de cardinais infinitos?

Essas e outras perguntas, além de relevantes, são muito importantes no estudo da Computação e Informática. Entretanto, um estudo mais completo ou aprofundado do assunto foge um pouco do escopo deste livro. Assim, este capítulo aborda a questão com ênfase nos conceitos, resultados e interpretações mais usados em Computação e Informática, sem deixar de lado a precisão formal.

Uma conseqüência importante do estudo da cardinalidade na Computação e Informática é que, computacionalmente falando, existem mais problemas não-solucionáveis do que problemas solucionáveis. Tal resultado é baseado no cardinal de todos os *problemas solucionáveis*, ou seja, problemas para os quais existe pelo menos um algoritmo (uma *Máquina de Turing*) capaz de solucioná-lo. De fato, o conjunto de todos os problemas solucionáveis está diretamente relacionado com o conceito de *discreto* (em oposição ao termo *contínuo*), justificando a denominação *Matemática Discreta*. Embora a noção intuitiva e o modelo formal da *Máquina de Turing* tenham sido apresentados em uma seção de leitura complementar, sua leitura é fortemente recomendada.

7.1 Cardinalidade Finita e Infinita

Cardinalidade é definida usando funções bijetoras. É interessante observar que o uso da bijeção é intuitivo e comum em diferentes civilizações e em todas as épocas. Por exemplo:

- para definir a cardinalidade de conjuntos, era usual fazer uma bijeção entre o conjunto de objetos em questão (por exemplo, um pequeno rebanho de ovelhas) com um subconjunto de dedos das mãos;
- quando os dedos da mão não eram suficientes para expressar uma cardinalidade maior, era usado, por exemplo, um conjunto de pedras ou de nós em cordas para estabelecer a bijeção;
- observe que se trata de uma bijeção qualquer, pois não existe uma definição de qual dedo (pedra ou nó) corresponde a qual objeto. Ou seja, a preocupação era saber se *existe* uma bijeção;
- claramente, se sobra (respectivamente, falta) um dedo (ou pedra ou nó), então falta (respectivamente, sobra) um objeto.

Definição 7.1 - Cardinalidade Finita, Cardinalidade Infinita
A *Cardinalidade* de um conjunto A, representada por:

$$\#A$$

é como segue:

a) *Finita* se existe uma bijeção entre A e o conjunto $\{1, 2, 3, \ldots, n\}$, para algum $n \in \mathbf{N}$. Nesse caso, afirma-se que (como fica o caso em que $n = 0$?):

$$\#A = n$$

b) *Infinita* se existe uma bijeção entre A e um subconjunto próprio de A. ❑

Portanto, um conjunto A é um:

- *conjunto finito* (ou seja, possui uma cardinalidade finita) se for possível representá-lo por extensão, como já introduzido;
- *conjunto infinito* se for possível retirar alguns elementos de A e, mesmo assim, estabelecer uma bijeção com A.

EXEMPLO 7.1 - Cardinalidade Infinita do Conjunto dos Números Inteiros
A função $f: \mathbf{Z} \rightarrow \mathbf{N}$ tal que, para qualquer $a \in \mathbf{Z}$:

se $a \geq 0$, então $f(a) = 2a$ — não-negativos são associados aos pares
se $a < 0$, então $f(a) = |2a| - 1$ — negativos são associados aos ímpares

onde $|2a|$ é o módulo (valor absoluto) de $2a$, é bijetora (por quê?). Claramente, **N** é subconjunto próprio de **Z**. Logo, **Z** é infinito. ❑

7.2 Conjunto Contável e Não-Contável

É importante destacar que nem todos os conjuntos infinitos possuem a mesma cardinalidade, o que contradiz a noção intuitiva da maioria das pessoas.

Definição 7.2 - Conjunto Contável, Conjunto Não-Contável, Conjunto Enumerável
Um conjunto A é dito:

a) *Finitamente Contável* se for finito;
b) *Infinitamente Contável* ou *Enumerável* se existe uma bijeção entre o conjunto A e um subconjunto infinito de **N**. Neste caso, a bijeção é denominada *enumeração* de A;
c) *Não-Contável*, caso contrário. ❑

O termo *conjunto contável* usualmente significa finitamente ou infinitamente contável. Observe que um conjunto é infinitamente contável se for possível enumerar seus elementos como uma seqüência infinita:

$$\langle a_1, a_2, a_3, \ldots \rangle$$

EXEMPLO 7.2 - Conjunto Contável
Os seguintes conjuntos são (infinitamente) contáveis ou enumeráveis:

Z (ver a função bijetora do EXEMPLO 7.1 - Cardinalidade Infinita do Conjunto dos Números Inteiros);
Q (prova sugerida como exercício). ❑

A prova de que um conjunto é não-contável pode ser realizada usando o método da *Diagonalização de Cantor* proposta por Georg Cantor (1845-1918). Esse método é freqüentemente aplicado em provas no contexto da Computação e Informática.

Teorema 7.3 - Conjunto Não-Contável
O seguinte conjunto é não-contável:

$$S = \{x \in \mathbf{R} \mid 0 < x < 1\}$$

Prova: *(por absurdo - Diagonalização de Cantor)*
Suponha que S é contável. Então, existe uma enumeração de S e, portanto, seus elementos podem ser enumerados em uma seqüência infinita como segue:

$$\langle s_1, s_2, s_3, \ldots \rangle$$

Claramente, qualquer número $s \in S$ pode ser representado como uma seqüência infinita de decimais, ou seja, dada por uma seqüência infinitamente contável de dígitos como segue:

$$s = 0{,}d_1\, d_2\, d_2 \ldots d_n \ldots$$

Por exemplo, $\pi / 10 = 0{,}31415\ldots$ Portanto, os componentes de $\langle s_1, s_2, s_3, \ldots \rangle$ podem ser representados da seguinte forma (observe a diagonal destacada):

$$\begin{aligned}
s_1 &= 0{,}\mathbf{d_{11}}\, d_{12}\, d_{13} \ldots d_{1n} \ldots \\
s_2 &= 0{,}d_{21}\, \mathbf{d_{22}}\, d_{23} \ldots d_{2n} \ldots \\
s_3 &= 0{,}d_{31}\, d_{32}\, \mathbf{d_{33}} \ldots d_{3n} \ldots \\
&\ldots \\
s_n &= 0{,}d_{n1}\, d_{n2}\, d_{n3} \ldots \mathbf{d_{nn}} \ldots \\
&\ldots
\end{aligned}$$

Seja r um número real construído como segue:

$$r = 0{,}e_1\, e_2\, e_3 \ldots e_n \ldots$$

sendo que, para $i \in \{1, 2, 3, \ldots\}$, a componente e_i é construída a partir da diagonal como segue:

$e_i = 1$, caso $d_{ii} \neq 1$
$e_i = 2$, caso $d_{ii} = 1$

Claramente, $r \in S$. Entretanto, r é diferente de qualquer número de $\langle s_1, s_2, s_3, \ldots \rangle$ pois difere pelo menos no dígito destacado na diagonal. Portanto, $r \notin S$, o que é uma contradição!

Logo, é um absurdo supor que S é contável. Portanto, S é não-contável. ❑

Neste momento, a prova de que o conjunto dos números reais é não-contável é simples. Antes considere a seguinte definição.

Definição 7.4 - Conjuntos Equipotentes
Dois conjuntos A e B são ditos *Equipotentes* quando existe uma função bijetora entre A e B. ❑

Logo, conjuntos equipotentes são aqueles que têm a mesma cardinalidade. Portanto:

- todos conjuntos enumeráveis são equipotentes;
- todo conjunto indexado é equipotente ao seu conjunto de índices.

Teorema 7.5 - R é um Conjunto Não-Contável
O conjunto dos números reais **R** é não-contável.

Prova: (*direta*)
Basta provar que **R** é equipotente a $S = \{x \in \mathbf{R} \mid 0 < x < 1\}$. De fato, seja $f: S \to \mathbf{R}$ tal que:

se $0 < s \leq 1/2$, então $f(s) = (1/2s) - 1$
se $1/2 \leq s < 1$, então $f(s) = (1/(2s-2)) + 1$

a qual é uma bijeção (tal verificação é sugerida como exercício). ❑

EXEMPLO 7.3 - Conjunto Não-Contável
Os seguintes conjuntos são não-contáveis:

I (conjuntos dos números irracionais);
C (conjunto dos números complexos);
$\mathbf{R}^2$ ❑

7.3 Cardinalidade dos Conjuntos Não-Contáveis

O que dizer sobre a cardinalidade dos conjuntos não-contáveis? Terão todos a mesma "quantidade" de elementos? A resposta é *não*, ou seja:

nem todos os conjuntos não-contáveis têm a mesma cardinalidade

Estendendo o conceito de contagem ao infinito, pode-se dizer que um conjunto A tem, pelo menos, tantos elementos quanto um conjunto B, ou seja, que:

$$\#A \leq \#B$$

quando existe uma função injetora $f: A \rightarrowtail B$. Para que a relação $\leq$ entre as cardinalidades seja uma relação de ordem parcial, ela deve ser:

- *reflexiva*, ou seja, $\#A \leq \#A$. De fato, basta considerar a função identidade $id_A: A \to A$, a qual é injetora;
- *transitiva*, ou seja, se $\#A \leq \#B$ e $\#B \leq \#C$, então $\#A \leq \#C$. De fato, já foi visto que a composição de funções injetoras é uma função injetora;
- *anti-simétrica*, ou seja, se $\#A \leq \#B$ e $\#B \leq \#A$, então $\#A = \#B$. Isso é garantido pelo *Teorema de Schröder-Bernstein*, enunciado a seguir, o qual não será demonstrado.

Teorema 7.6 - Schröder-Bernstein
Sejam A e B dois conjuntos tais que existem duas funções injetoras:

$$f_1: A \rightarrowtail B \quad \text{e} \quad f_2: B \rightarrowtail A$$

Então existe uma função bijetora:

$$g: A \leftrightarrow B$$ ❑

Um resultado que garante que existem cardinais não-contáveis em quantidade infinita é o fato de que o conjunto das partes de um conjunto tem sempre cardinalidade maior que este. Esse fato é conhecido como *Teorema de Cantor*, apresentado a seguir. Na demonstração, lembre que:

$$n < m \Leftrightarrow n \leq m \wedge n \neq m$$

Teorema 7.7 - Cantor
Seja C um conjunto e 2^C o conjunto das partes de C. Então:

$$\#C < \#2^C$$

Prova:
A prova é dividida em duas partes:

- para mostrar que $\#C \leq \#2^C$, basta apresentar uma função injetora $f: C \rightarrowtail 2^C$
- para mostra que $\#C \neq \#2^C$, basta mostrar que *não* existe uma função bijetora $g: C \leftrightarrow 2^C$

Parte 1: (*direta*)
Seja $f: C \to 2^C$ uma função tal que, para todo $s \in C$, vale:

$$f(s) = \{s\}$$

Claramente f é injetora (por quê?). Portanto, $\#C \leq \#2^C$.

Parte 2: (*por absurdo*)
Suponha que existe uma função bijetora $g: C \leftrightarrow 2^C$. Seja o seguinte subconjunto A de C:

$$A = \{a \in C \mid a \notin g(a)\}$$

Como g é uma função bijetora, em particular, é uma função sobrejetora. Portanto existe $c \in C$ tal que $g(c) = A$. Assim, existem dois casos:

Caso 1: $c \in A$
$c \in A \Rightarrow$ $g(c) = A$
$c \in g(c) \Rightarrow$ pela definição de A
$c \notin A$

Caso 2: $c \notin A$
$c \notin A \Rightarrow$ $g(c) = A$
$c \notin g(c) \Rightarrow$ pela definição de A
$c \in A$

o que é uma *contradição* (um elemento não pode pertencer e não pertencer a um conjunto ao mesmo tempo)! Ou seja, é absurdo supor que existe tal função bijetora. Logo, *não* existe uma função bijetora entre C e 2^C. Portanto, $\#C \neq \#2^C$. ❑

A partir desse resultado, pode-se pensar em uma classe infinita de números cardinais.

Definição 7.8 - Cardinal
Um *Cardinal* é uma classe de equivalência de conjuntos equipotentes. ❑

Dessa forma, a classe dos cardinais é a classe de todas as classes de equivalência dos conjuntos equipotentes. Observe a semelhança com o Paradoxo de Russell se a classe de todos os cardinais for tomada como um cardinal. Portanto, tal classe *não* é um conjunto.

Em geral utiliza-se a primeira letra do alfabeto hebraico $\aleph$ (lê-se "alef") com índices para indicar cardinais infinitos conhecidos. De especial interesse para Ciência da Computação é o cardinal do conjunto dos números naturais **N**, denotado por $\aleph_0$. Assim, o cardinal de qualquer conjunto contável (infinito) é $\aleph_0$. De fato:

$\aleph_0$ é o menor cardinal dos conjuntos infinitos

Define-se $\aleph_{i+1}$ como sendo o menor cardinal maior que $\aleph_i$. Prova-se que o conjunto das partes de **N** é equipotente ao conjunto **R**. Considerando que 2^k denota o cardinal do conjunto das partes de conjuntos com cardinalidade k, pode-se afirmar que $2^{\aleph_0}$ é a cardinalidade do conjunto dos números reais, ou seja, é a cardinalidade do *continuum*.

Entretanto, resta a questão se $\aleph_{i+1} = 2^{\aleph_i}$. Essa asserção é conhecida como *Hipótese do Continuum* e foi formulada como hipótese por Cantor uma vez que ele não conseguiu obter sua prova a partir dos axiomas que estabeleceu para a sua *Teoria dos Conjuntos*. Atualmente, primeiro devido a Gödel (década de 1930) e depois a Cohen (década de 1950), sabe-se que esse fato é independente da Teoria dos Conjuntos, ou seja, tanto a Hipótese do *Continuum*, quanto sua negação são consistentes com o restante da Teoria dos Conjuntos, não gerando nenhuma contradição à adição de cada uma em separado à mesma.

7.4 Cardinal do Conjunto de Todos os Problemas Solucionáveis

O estudo dos cardinais é de fundamental importância em Computação e Informática, com especial interesse no estudo dos problemas que podem ser resolvidos usando um sistema computador. De fato, prova-se que existem $\aleph_0$ programas que podem ser definidos em qualquer linguagem de programação de propósitos gerais (como a linguagem Pascal ou a linguagem C). Como, por exemplo, existem $2^{\aleph_0}$ funções de **N** para **N**, conclui-se que existem infinitas funções que não podem ser representadas algoritmicamente, ou seja, que não são computáveis (em um sistema computador). Portanto, pode-se afirmar que:

- existem infinitos, mas contáveis, *problemas solucionáveis*, ou seja, problemas para os quais existe pelo menos um algoritmo (uma *Máquina de Turing*) capaz de solucioná-lo;
- existem infinitos, e não-contáveis, *problemas não-solucionáveis*, ou seja, para os quais não existe qualquer algoritmo (programa) capaz de solucioná-los.

Adicionalmente, como existe uma função (injetora) de inclusão de **N** para **R**, e não existe função injetora de **R** para **N**, pode-se afirmar que $\#\mathbf{N} < \#\mathbf{R}$. Portanto, o cardinal dos problemas não-solucionáveis é maior que o cardinal dos problemas solucionáveis.

7.5 Leitura Complementar: Máquina de Turing

Quando da introdução do Autômato Finito no Capítulo 6 - Funções Parciais e Totais, foi destacada a limitação da capacidade de solucionar problemas deste modelo. Como ilustração, prova-se que não existe autômato finito capaz de reconhecer construções tão simples como palíndromos de qualquer tamanho, qualquer número de parênteses aninhados ou balanceados (como em uma expressão aritmética), etc.

No Capítulo 2 - Lógica e Técnicas de Demonstração, foi dito que a Máquina de Turing é aceita como uma formalização do conceito de algoritmo computável, e conseqüentemente, do que é possível solucionar em um sistema computador. No Capítulo 3 - Álgebra de Conjuntos, algumas questões relacionadas com a capacidade de solucionar problemas foram discutidas.

Entretanto, o conceito de Máquina de Turing não foi introduzido. Adicionalmente, a seguinte pergunta destaca-se: qual a diferença fundamental entre os dois modelos, de tal forma que o poder computacional da Máquina de Turing seja tão maior que o do Autômato Finito?

A resposta a essa questão ilustra o porquê de a *Matemática Discreta* possuir como ênfase os estudos matemáticos baseados em conjuntos contáveis, *finitos* ou *infinitos* (conceito apresentado no Capítulo 1 - Introdução e Conceitos Básicos). De fato:

a) A definição de *Autômato Finito*, como o próprio nome indica, é baseada na noção de estados *finitos* e *predefinidos* que o sistema pode assumir. Portanto, é baseada na noção de conjunto *finitamente contável*;
b) Em contrapartida, a definição de *Máquina de Turing* é baseada na noção de estados possíveis finitos, mas *não-predefinidos*, o que implica uma noção de "tão grande quanto necessário". Portanto, é baseada na noção de conjunto *infinitamente contável*.

7.5.1 Noção Intuitiva da Máquina de Turing

A *Máquina de Turing*, proposta por Alan Turing em 1936, é um mecanismo simples que formaliza a idéia de uma pessoa que realiza cálculos. Lembra, em muito, os computadores atuais, embora tenha sido proposta anos antes do primeiro computador digital. Apesar de sua simplicidade, o modelo Máquina de Turing possui, no mínimo, o mesmo poder computacional de qualquer computador de propósito geral.

O ponto de partida de Turing foi analisar a situação na qual uma pessoa, equipada com um instrumento de escrita e um apagador, realiza cálculos em uma folha de papel, organizada em quadrados.

Inicialmente, suponha que a folha de papel contém somente os dados iniciais do problema. O trabalho da pessoa pode ser resumido em seqüências de operações simples como segue:

- ler um símbolo de um quadrado;
- alterar um símbolo em um quadrado;
- mover os olhos para outro quadrado.

Quando é encontrada alguma representação satisfatória para a resposta desejada, a pessoa termina seus cálculos. Para viabilizar esse procedimento, as seguintes hipóteses são aceitáveis:

- a natureza bidimensional do papel não é um requerimento essencial para os cálculos. Pode ser assumido que o papel consiste de uma fita infinita organizada em quadrados;
- o conjunto de símbolos pode ser finito, pois se pode utilizar seqüências de símbolos;
- o conjunto de estados da mente da pessoa durante o processo de cálculo é finito. Mais ainda, entre esses estados, existem dois em particular: "estado inicial" e "estado final", correspondendo ao início e ao fim dos cálculos, respectivamente;
- o comportamento da pessoa, a cada momento, é determinado somente pelo seu estado presente e pelo símbolo para o qual sua atenção está voltada;
- a pessoa é capaz de observar e alterar o símbolo de apenas um quadrado de cada vez, bem como de transferir sua atenção somente para um dos quadrados adjacentes.

7.5.2 Modelo e Exemplo

Essa noção de uma pessoa calculando pode ser vista como uma máquina, constituída de três partes, como segue:

a) *Fita*. Usada simultaneamente como dispositivo de entrada, de saída e de memória de trabalho;
b) *Unidade de Controle*. Reflete o estado corrente da máquina. Possui uma unidade de leitura e gravação (cabeça da fita), a qual acessa uma célula da fita de cada vez e movimenta-se para a esquerda ou para a direita;
c) *Programa* ou *Função de Transição*. Função que define o estado da máquina e comanda as leituras, as gravações e o sentido de movimento da cabeça.

A fita é finita à esquerda e *infinita* (tão grande quanto necessário) à direita, sendo dividida em células, cada uma armazenando um símbolo. Os símbolos podem pertencer ao alfabeto de entrada, ao alfabeto auxiliar ou ainda ser "branco" ou "marcador de início de fita". Inicialmente, a palavra a ser processada (ou seja, a informação de entrada para a máquina) ocupa as células mais à esquerda após o marcador de início de fita, ficando as demais com "branco", como ilustrado na Figura 7.1, onde β e ⊙ representam "branco" e "marcador de início de fita", respectivamente.

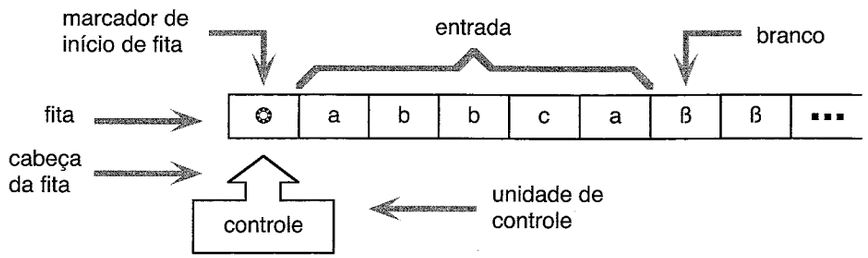


Figura 7.1 Fita e unidade de controle de uma Máquina de Turing

A unidade de controle possui um número *finito* e *predefinido* de estados. A *cabeça da fita* lê o símbolo de uma célula de cada vez e grava um novo símbolo. Após a leitura/gravação (a gravação é realizada na mesma célula de leitura), a cabeça move uma célula para a direita ou para a esquerda. O símbolo gravado e o sentido do movimento são definidos pelo programa.

O programa é uma função que, dependendo do estado corrente da máquina e do símbolo lido, determina o símbolo a ser gravado, o sentido do movimento da cabeça e o novo estado.

EXEMPLO 7.4 - Máquina de Turing - Duplo Balanceamento

Considere a Máquina de Turing ilustrada na Figura 7.2, definida sobre o alfabeto $\Sigma = \{a, b\}$ e tendo como alfabeto auxiliar (de trabalho) $\{A, B\}$, na qual:

- nodos representam *estados* da máquina, os quais são em número *finito*;
- arcos representam *transições* ou *computações atômicas*, as quais são em número *finito*. O conjunto de todas as transições constitui o programa da Máquina de Turing;
- o estado q_0 é dito *estado inicial* e é representado de forma diferenciada (no caso, o nodo destino de uma seta sem origem);
- o estado q_4 é dito *estado final* e é representado de forma diferenciada (no caso, o traço da circunferência do nodo é mais forte). ❑

A função programa considera o estado corrente e o símbolo lido da fita para determinar o novo estado, o símbolo a ser gravado e o sentido de movimento da cabeça, onde esquerda e direita são representados por E e D, respectivamente. A interpretação como um grafo é ilustrada na Figura 7.3.

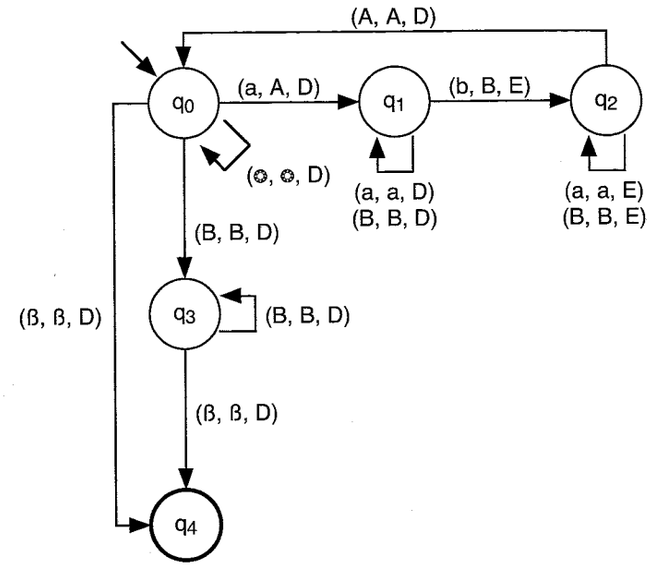


Figura 7.2 Máquina de Turing

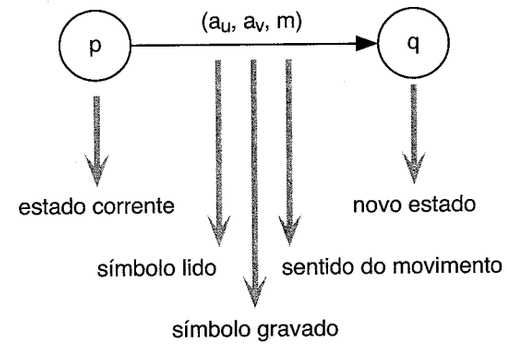


Figura 7.3 Interpretação da função programa como um grafo

O processamento de uma Máquina de Turing, para uma palavra de entrada w armazenada na fita, consiste na sucessiva aplicação da função programa a partir do estado inicial q_0 e da cabeça posicionada na célula mais à esquerda da fita até ocorrer uma condição de parada. O processamento de M para a entrada w pode parar ou ficar em *loop* infinito. A parada pode ser de duas maneiras: aceitando ou rejeitando a entrada w. As condições de parada são as seguintes:

a) A máquina assume um estado final: a máquina pára e a palavra de entrada é aceita;
b) A função programa é indefinida para o argumento (símbolo lido e estado corrente): a máquina pára e a palavra de entrada é rejeitada;
c) O argumento corrente da função programa define um movimento à esquerda e a cabeça da fita já se encontra na célula mais à esquerda: a máquina pára e a palavra de entrada é rejeitada.

A Máquina de Turing ilustrada na Figura 7.2 aceita a seguinte linguagem definida sobre o alfabeto $\Sigma = \{a, b\}$, conhecida como "duplo balanceamento":

$$L_1 = \{\varepsilon, ab, aabb, aaabbb, \ldots\}$$

Essa linguagem é um exemplo clássico e de fundamental importância no estudo das linguagens, pois permite estabelecer uma analogia com linguagens que possuem duplo balanceamento em sua estrutura como, por exemplo:

a) *Linguagens Bloco-Estruturadas*, como a linguagem de programação Pascal, na qual cada bloco é determinado pelo trecho de programa contido entre as palavras `begin` e `end`;
b) Linguagens com parênteses balanceados, ou seja, um número qualquer de parênteses em um texto, eventualmente encadeados (aninhados), de tal forma a garantir que, para cada parêntese aberto (respectivamente, fechado), existe um correspondente parêntese fechado (respectivamente, aberto), como as expressões aritméticas, presentes na maioria das linguagens de programação.

Observe que, analogamente ao autômato finito, o programa de uma Máquina de Turing pode ser definido como uma função parcial. De fato, supondo um conjunto finito de estados Q e um conjunto de símbolos S (incluindo o alfabeto Σ, os símbolos especiais β e ⊙ e o alfabeto auxiliar), a *função programa* pode ser definida como segue:

$$\delta: Q \times S \rightarrow Q \times S \times \{E, D\}$$

Para a Máquina de Turing da Figura 7.2, a correspondente função programa é representada na forma de matriz na Figura 7.4, na qual cada estado (linha) e símbolo lido (coluna) determina um novo estado, o símbolo gravado e movimento da cabeça (célula da matriz).

δ	⊙	a	b	A	B	β
q_0	$(q_0, ⊙, D)$	(q_1, A, D)			(q_3, B, D)	(q_4, β, D)
q_1		(q_1, a, D)	(q_2, B, E)		(q_1, B, D)	
q_2		(q_2, a, E)		(q_0, A, D)	(q_2, B, E)	
q_3					(q_3, B, D)	(q_4, β, D)
q_4						

Figura 7.4 Máquina de Turing - função programa na forma de matriz

O algoritmo apresentado reconhece o primeiro símbolo a, o qual é marcado como A, e movimenta a cabeça da fita à direita, procurando o b correspondente, o qual é marcado como B. Esse ciclo é repetido sucessivamente até identificar, para cada a, o seu correspondente b. Adicionalmente, o algoritmo garante que qualquer outra palavra que não esteja na forma de um duplo balanceamento é rejeitada. A Figura 7.5 ilustra a seqüência do processamento da Máquina de Turing em questão para a entrada $w = aabb$.

Observação 7.9 - Máquina de Turing × Algoritmo

Ao longo de todo o livro, tem sido afirmado que a *Máquina de Turing* é aceita como uma formalização do conceito de *algoritmo*. Entretanto, também é usual considerar que o conceito de algoritmo corresponde a uma Máquina de Turing que sempre pára para qualquer entrada. Nesse caso, uma máquina que eventualmente fica processando indefinidamente (em *loop* infinito) não seria considerada um algoritmo. ❑

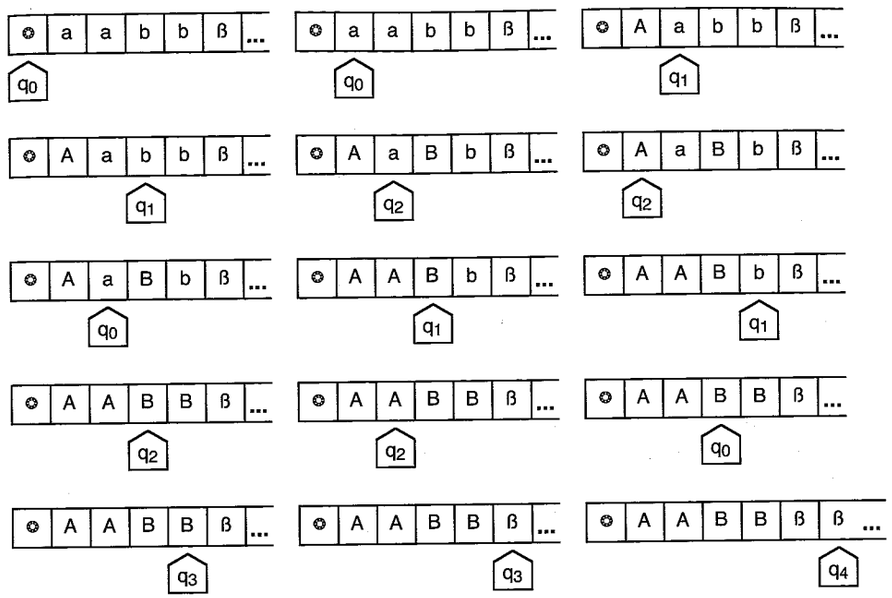


Figura 7.5 Seqüência de processamento de uma Máquina de Turing

7.5.3 Cardinal do Conjunto de Todas as Máquinas de Turing

Já foi afirmado que existem infinitos, mas contáveis, problemas solucionáveis, ou seja, problemas para os quais existe pelo menos uma Máquina de Turing capaz de solucioná-lo. De fato, o conjunto de todas as Máquinas de Turing Δ é equipotente ao conjunto dos números naturais. A prova pode ser facilmente entendida como segue (o detalhamento é sugerido como exercício e pode ser encontrado na maioria dos livros de Teoria da Computação):

- dado que uma função programa de uma máquina de Turing é tal que
$$\delta \subseteq (Q \times S) \times (Q \times S \times \{E, D\})$$
- então, cada par de δ possui como primeira componente um par (estado e símbolo) e como segunda componente uma terna (estado, símbolo e sentido do movimento);
- portanto, cada par de δ contém um total de cinco informações, todas com possibilidades finitas de valores;
- logo, supondo que o cardinal de δ é n (uma função programa sempre é finita), $5n$ componentes definem toda a função programa;
- pelo *Teorema Fundamental da Aritmética*, sabe-se que cada número natural é univocamente decomposto em seus fatores primos;
- assim, as $5n$ componentes de δ podem ser codificadas univocamente como um número natural. Como, para diferentes funções programas, tem-se diferentes números naturais, essa construção caracteriza uma função injetora de Δ para $\mathbf{N}$.

Portanto, $\#\Delta \leq \#\mathbf{N}$. Como $\#\Delta$ é infinito, e sabendo-se que $\#\mathbf{N} = \aleph_0$ é o menor cardinal dos conjuntos infinitos, então Δ e $\mathbf{N}$ são conjuntos equipotentes.

7.6 Exercícios

Exercício 7.1 Afirma-se que a cardinalidade de um conjunto é finita, se existe uma bijeção entre A e o conjunto $N = \{1, 2, 3, \ldots, n\}$, para algum $n \in \mathbf{N}$. Supondo $n = 0$:

a) Qual é o conjunto A?

b) Qual é a correspondente função bijetora?

Exercício 7.2 Prove que, de fato, a seguinte função é bijetora:

$f: \mathbf{Z} \rightarrow \mathbf{N}$ tal que:

se $a \geq 0$, então $f(a) = 2a$ — positivos são associados aos pares

se $a < 0$, então $f(a) = |2a| - 1$ — negativos são associados aos ímpares

Exercício 7.3 Prove que $\mathbf{R}$ é um conjunto infinito, apresentando uma bijeção com um subconjunto próprio de $\mathbf{R}$.

Exercício 7.4 Prove que $\mathbf{Q}$ é contável.

Sugestão: considere $\mathbf{Q}$ como sendo um conjunto de pares e, para construir uma função injetora de $\mathbf{Q}$ para $\mathbf{N}$, lembre-se de que a decomposição de um número em seus fatores primos é única.

Exercício 7.5 Prove que a função $f: S \rightarrow \mathbf{R}$ apresentada na prova do Teorema 7.5 - $\mathbf{R}$ é um Conjunto Não-Contável é de fato uma bijeção.

Dica: na investigação se f possui inversa, considere a seguinte função $g: \mathbf{R} \rightarrow S$ tal que:

se $x \geq 0$, então $g(x) = 1 / (2x + 2)$

se $x < 0$, então $g(x) = (1 / (2x - 2)) + 1$

Exercício 7.6 Prove que:

a) Nem sempre a intersecção de conjuntos não-contáveis é não-contável;

b) Nem sempre a diferença de conjuntos não-contáveis é não-contável.

Exercício 7.7 Prove que:

a) A união de conjuntos contáveis é contável;

b) O produto cartesiano de conjuntos contáveis é contável.

Exercício 7.8 Prove que, se $A \subseteq B$ e A é infinito, então B é infinito.

Exercício 7.9 Lembre-se de que, se existe uma função injetora $f: A \rightarrowtail B$, então $\#A \leq \#B$. Sabendo que $\mathbf{R}$ é não-contável, prove que são conjuntos não-contáveis:

a) $\mathbf{R}^2$

b) $\mathbf{N} \times \mathbf{R}$

c) $\mathbf{C}$ (conjunto dos números complexos)

Exercício 7.10 Suponha o alfabeto $\Sigma = \{a, b\}$. Desenvolva Máquinas de Turing, determinísticas ou não, que aceitem as seguintes linguagens:

a) $\{\varepsilon, a, b\}$

b) $\{w \in \Sigma^* \mid w$ tem o mesmo número de símbolos a e $b\}$

c) $\{w \in \Sigma^* \mid$ o décimo símbolo da direita para a esquerda de w é $a\}$

d) $\{waw \mid w \in \Sigma^*\}$

Exercício 7.11 Detalhe a prova de que o conjunto de todas as Máquinas de Turing é equipotente ao conjunto dos números naturais.

8 Indução e Recursão

8.1 Princípio da Indução Matemática

O *Princípio da Indução Matemática* é uma técnica para lidar com tipos de dados que têm uma *relação de boa-ordem*, isto é, uma relação onde todo subconjunto não-vazio do tipo de dado tem um elemento mínimo segundo essa relação de ordem. Um exemplo típico é o conjunto dos números naturais (na realidade um tipo de dado). Dada uma boa ordem, pode-se aplicar indução para provar propriedades que valem para todo elemento do tipo de dado.

Por simplicidade, o tipo de dados que tem uma relação de boa-ordem considerado é o conjunto dos números naturais $\mathbf{N}$ (ou qualquer outro conjunto isomorfo a $\mathbf{N}$).

Um exemplo simples que ilustra o Princípio da Indução Matemática é o *efeito dominó* (veja a Figura 8.1): uma fila sem fim de peças do jogo dominó para a qual, ao derrubar a primeira peça, todas as demais peças são derrubadas em cadeia. Para estar certo de que tal fato ocorre, suponha que são verdadeiras as seguintes proposições:

a) a primeira peça é derrubada na direção das demais;

b) se qualquer peça está suficientemente próxima da seguinte na fila, então, ao ser derrubada, fará com que a sua vizinha seguinte também seja derrubada.

Então:

- pelo item a), a primeira peça é derrubada;
- pelo item b), a segunda peça é derrubada;
- pelo item b), a terceira peça é derrubada;
- pelo item b), a quarta peça é derrubada;
- e assim sucessivamente.

Portanto, para i tão grande quanto se queira, pode-se afirmar que a i-ésima peça é derrubada. Logo, para qualquer n, pode-se afirmar que a n-ésima peça é derrubada.

O Princípio da Indução Matemática pode ser resumido como segue:

se o início é correto e se coisa alguma pode dar errada,
então sempre será correto

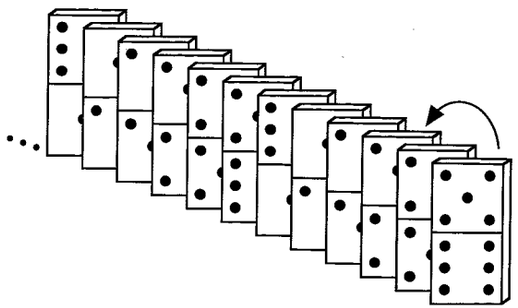


Figura 8.1 Efeito dominó

Definição 8.1 - Princípio da Indução Matemática

Seja $p(n)$ uma proposição sobre $M = \{n \in \mathbf{N} \mid n \geq m \text{ e } m \in \mathbf{N}\}$. O *Princípio da Indução Matemática* é como segue:

a) $p(m)$ é verdadeira;

b) Para qualquer $k \in M$, vale $p(k) \Rightarrow p(k+1)$

c) Então, para qualquer $n \in M$, $p(n)$ é verdadeira.

Nesse caso, utiliza-se a seguinte denominação:

- *base de indução*: a proposição $p(m)$
- *hipótese de indução*: a proposição $p(k)$
- *passo de indução*: a implicação $p(k) \Rightarrow p(k+1)$ ❑

Na definição acima, o Princípio da Indução Matemática foi apresentado na sua forma mais tradicional, usualmente denominada de *Primeiro Princípio da Indução Matemática*. Outras formulações alternativas do mesmo princípio são apresentadas adiante. Adicionalmente, observe que, embora o Princípio da Indução Matemática seja definido considerando o conjunto dos números naturais, qualquer conjunto isomorfo a $\mathbf{N}$ pode ser considerado.

Duas aplicações do Princípio da Indução Matemática destacam-se:

a) *Prova Indutiva* ou *Prova por Indução*, uma técnica de demonstração muito comum no contexto da Computação e Informática a qual não é do domínio da lógica pura;

b) *Definição Indutiva* ou *Definição Recursiva* igualmente comum no contexto da Computação e Informática, a qual já foi usada anteriormente de maneira informal.

Um conceito próximo ao de indução e presente na grande maioria das linguagens de programação é o de *recursão*.

Neste capítulo, são discutidas diversas construções baseadas em indução e recursão e de especial interesse para Computação e Informática como, por exemplo:

- *Computações* de um autômato finito;
- *Gramáticas de Chomsky* e a *Forma de Backus Naur* (ou *BNF*);
- *Expressões Regulares*;
- *Funções Recursivas Parciais* ou *Funções Recursivas de Kleene*.

8.2 Prova Indutiva

A *Prova Indutiva* ou *Prova por Indução* é uma técnica de demonstração baseada no Princípio da Indução Matemática a qual não é do domínio da lógica pura. Trata-se de uma técnica que se limita a confirmar se uma determinada conjectura $p(n)$ sobre $n \in M = \{n \in \mathbf{N} \mid n \geq m \text{ e } m \in \mathbf{N}\}$ é correta.

Assim, em uma demonstração por indução, deve-se demonstrar a base de indução $p(m)$ e, tendo fixado um k, supor verdadeira a hipótese de indução $p(k)$ e demonstrar o passo de indução, ou seja, que $p(k) \rightarrow p(k+1)$ é, de fato, uma implicação.

EXEMPLO 8.1 - Prova por Indução: $p(n): n < 2^n$

Considere o seguinte teorema:

para qualquer $n \in \mathbf{N}$, *vale* $n < 2^n$

Uma prova por indução de $p(n): n < 2^n$ é como segue:

a) *Base de Indução*. Seja $k = 0$. Então:

$$0 < 1 = 2^0$$

Portanto, $p(0)$ é verdadeira;

b) *Hipótese de Indução*. Suponha que, para algum $k \in \mathbf{N}$, tem-se que:

$$p(k): k < 2^k \text{ é verdadeira}$$

c) *Passo de Indução*. A prova para $p(k+1): k+1 < 2^{k+1}$ é como segue:

$k+1 <$ pela hipótese de indução

$2^k + 1 \leq 2^k + 2^k = 2 * 2^k = 2^{k+1}$

Logo, para qualquer $n \in \mathbf{N}$, tem-se que $n < 2^n$. ❑

EXEMPLO 8.2 - Prova por Indução: $p(n): 2^n < n!$

Considere o seguinte teorema:

para qualquer $n \in \mathbf{N}$, *se* $n > 3$, *então* $2^n < n!$

Uma prova por indução de $p(n): 2^n < n!$ é como segue:

a) *Base de Indução*. Seja $k = 4$. Então:

$$2^4 = 16 < 24 = 4!$$

Portanto, $p(4)$ é verdadeira;

b) *Hipótese de Indução*. Suponha que, para algum $k \in \mathbf{N}$ tal que $k > 3$:

$$p(k): 2^k < k! \text{ é verdadeira}$$

c) *Passo de Indução*. A prova para $p(k+1): 2^{k+1} < (k+1)!$ é como segue:

$2^{k+1} = 2 * 2^k <$ pela hipótese de indução

$2 * k! < (k+1) * k! = (k+1)!$

Logo, para qualquer $n \in \mathbf{N}$, se $n > 3$, então $2^n < n!$ ❑

EXEMPLO 8.3 - Prova por Indução: $1 + 2 + \ldots + n = (n^2 + n)/2$

Considere o seguinte teorema:

para qualquer $n \in \mathbf{N}$, *tem-se que* $1 + 2 + \ldots + n = (n^2 + n)/2$

Uma prova por indução de $p(n): 1 + 2 + \ldots + n = (n^2 + n)/2$ é como segue:

a) *Base de Indução*. Seja $k=0$. Então:

$$(0^2+0)/2 = (0+0)/2 = 0/2 = 0$$

Portanto, $p(0)$ é verdadeira. Note que:

$$1+2+\ldots+k = 0+1+2+\ldots+k$$

pois 0 é o elemento neutro da adição, e, portanto, o somatório até $k=0$ é perfeitamente definido;

b) *Hipótese de Indução*. Suponha que, para algum $k \in \mathbf{N}$:

$$p(k): 1+2+\ldots+k = (k^2+k)/2 \text{ é verdadeira}$$

c) *Passo de Indução*. Prova para $p(k+1): 1+2+\ldots+k+(k+1) = ((k+1)^2+k+1)/2$ é como segue:

$$\begin{aligned}
&1+2+\ldots+k+(k+1) = \\
&(1+2+\ldots+k)+(k+1) = && \text{pela hipótese de indução} \\
&(k^2+k)/2+(k+1) = \\
&(k^2+k)/2+(2k+2)/2 = \\
&(k^2+k+2k+2)/2 = \\
&((k^2+2k+1)+(k+1))/2 = \\
&((k+1)^2+(k+1))/2
\end{aligned}$$

Portanto, $p(k+1): 1+2+\ldots+k+(k+1) = ((k+1)^2+(k+1))/2$ é verdadeira.

Logo, para qualquer $n \in \mathbf{N}$, tem-se que $1+2+\ldots+n = (n^2+n)/2$ ❑

EXEMPLO 8.4 - Prova por Indução: $\#2^A = 2^{\#A}$

Quando da introdução da definição do conjunto das partes, foi afirmado que, se o cardinal de um conjunto A é n, então o cardinal de 2^A é 2^n, o que justifica a notação 2^A. Assim, considere o seguinte teorema (observe que se refere a conjuntos finitos):

para qualquer conjunto finito A, se $\#A = n$, então $\#2^A = 2^n$

Antes de apresentar uma prova por indução de $p(n): \#\mathbf{P}(A) = 2^n$, considere o seguinte exemplo motivacional:

$$\begin{aligned}
&\mathbf{P}(\varnothing) = \{\varnothing\} \\
&\mathbf{P}(\{a\}) = \{\varnothing, \{a\}\} \\
&\mathbf{P}(\{a,b\}) = \{\varnothing, \{a\}, \{b\}, \{a,b\}\} \\
&\mathbf{P}(\{a,b,c\}) = \{\varnothing, \{a\}, \{b\}, \{c\}, \{a,b\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}
\end{aligned}$$

Em particular, observe os conjuntos $\mathbf{P}(\{a,b\})$ e $\mathbf{P}(\{a,b,c\})$ (o mesmo raciocínio vale para os demais casos):

- a metade dos elementos de $\mathbf{P}(\{a,b,c\})$ corresponde a todos os elementos de $\mathbf{P}(\{a,b\})$, ou seja:

$$\mathbf{P}(\{a,b,c\}) = \mathbf{P}(\{a,b\}) \cup \{\{c\}, \{a,c\}, \{b,c\}, \{a,b,c\}\}$$

- e a outra metade dos elementos de $\mathbf{P}(\{a,b,c\})$ corresponde a todos os elementos de $\mathbf{P}(\{a,b\})$, adicionando-se o elemento c a cada conjunto, ou seja:

$$\mathbf{P}(\{a,b,c\}) = \mathbf{P}(\{a,b\}) \cup \{\varnothing \cup \{c\}, \{a\} \cup \{c\}, \{b\} \cup \{c\}, \{a,b\} \cup \{c\}\}$$

o que justifica o fato de que $\#\mathbf{P}(\{a,b,c\})$ possui o dobro de elementos de $\#\mathbf{P}(\{a,b\})$. Assim, a prova que segue mostra que:

cada elemento adicionado a um conjunto

duplica o cardinal do correspondente conjunto das partes

a) *Base de Indução*. Seja $k=0$ ($\#A=0$). Então $A=\varnothing$, e:

$$\mathbf{P}(\varnothing) = \{\varnothing\}$$

Portanto, $p(0)$ é verdadeira. Note-se que:

$$\#\mathbf{P}(\varnothing) = 1 = 2^0 = 2^{\#\varnothing}$$

b) *Hipótese de Indução*. Suponha que, para algum $k \in \mathbf{N}$ ($\#A=k$):

$$p(k): \#\mathbf{P}(A) = 2^{\#A} \text{ é verdadeira}$$

c) *Passo de Indução*. Prova para $p(k+1)$. Sem perda de generalidade, suponha os seguintes conjuntos (lembre-se de que os conjuntos com o mesmo cardinal são isomorfos):

$$\begin{aligned}
&A = \{1,2,3,\ldots,k\} \\
&B = \{1,2,3,\ldots,k,k+1\}
\end{aligned}$$

Por hipótese de indução, $\#\mathbf{P}(A) = 2^k$. Como $\mathbf{P}(A)$ são todos os subconjuntos de B que não possuem o elemento $k+1$, então os demais subconjuntos B são aqueles que contêm o elemento $k+1$, ou seja, a união de cada conjunto de $\mathbf{P}(A)$ com $\{k+1\}$, resultando em outros 2^k conjuntos. Portanto:

$$\#\mathbf{P}(B) = 2^k + 2^k = 2^{k+1}$$

Logo, para qualquer conjunto finito A, se $\#A=n$, então $\#2^A = 2^n$. ❑

8.3 Segundo Princípio da Indução Matemática

Em diferentes momentos, pode ser conveniente trabalhar com outras formulações do Princípio da Indução Matemática. Uma formulação especialmente importante para Computação e Informática é o *Segundo Princípio da Indução Matemática*.

Para facilitar o entendimento, inicialmente é reproduzido o princípio apresentado, o qual também é denominado de *Primeiro Princípio da Indução Matemática*:

Seja $p(n)$ uma proposição sobre $M = \{n \in \mathbf{N} \mid n \geq m \text{ e } m \in \mathbf{N}\}$. O Princípio da Indução Matemática é como segue:

a) $p(m)$ é verdadeira;
b) Para qualquer $k \in M$, $p(k) \Rightarrow p(k+1)$;
c) Então, para qualquer $n \in M$, $p(n)$ é verdadeira.

O segundo princípio considera não apenas o resultado anterior $p(k)$, mas todos os anteriores, para concluir $p(k+1)$. A seguir, são apresentadas duas versões equivalentes desse princípio.

Definição 8.2 - Segundo Princípio da Indução Matemática

Seja $p(n)$ uma proposição sobre $M = \{n \in \mathbf{N} \mid n \geq m \text{ e } m \in \mathbf{N}\}$. O *Segundo Princípio da Indução Matemática* pode, equivalentemente, ser definido como segue:

a) *Primeira Versão*.

a.1) $p(m)$ é verdadeira;
a.2) Para qualquer $k \in M$, vale:

$$p(m) \wedge p(m+1) \wedge \ldots \wedge p(k) \Rightarrow p(k+1)$$

a.3) Então, para qualquer $n \in M$, $p(n)$ é verdadeira.

b) *Segunda Versão*. Suponha $t \in \mathbf{N}$:

b.1) $p(m), p(m+1), \ldots, p(m+t)$, são verdadeiras;

b.2) Para qualquer $k \in M$ tal que $k \geq m+t$, vale:

$$p(m) \wedge p(m+1) \wedge \ldots \wedge p(k) \Rightarrow p(k+1)$$

b.3) Então, para qualquer $n \in \mathbf{N}$, $p(n)$ é verdadeira. ❑

Observe que a segunda versão do Segundo Princípio da Indução Matemática prova os t primeiros casos em separado para verificar a base de indução.

Uma aplicação usual desse princípio está na definição e na prova de propriedades de expressões, fórmulas, árvores, etc., razão pela qual esse princípio freqüentemente é denominado de *indução em estrutura*, *indução estruturada*, ou ainda, *indução estrutural*.

EXEMPLO 8.5 - Segundo Princípio da Indução Matemática: Proposição Lógica
Considere o seguinte teorema:

Suponha que p é uma proposição lógica
a qual contém exclusivamente os conetivos lógicos conjunção, disjunção e condição.
Se o valor-verdade de todos os átomos de p é V, então o valor-verdade de p é V

Uma prova por indução (no número de átomos, usando a primeira versão do Segundo Princípio da Indução) é como segue:

a) *Base de Indução*. Seja $k=1$. Então p é um átomo. Portanto, por hipótese, o valor-verdade de p é V;

b) *Hipótese de Indução*. Suponha que, para algum $k \in \mathbf{N}$, e para qualquer $u \in \mathbf{N}$ tal que $u \leq k$, se o número de átomos de p é u, então o valor-verdade de p é V;

c) *Passo de Indução*. Seja p uma proposição com $k+1$ átomos. Então p pode ser reescrita em um dos seguintes casos, sendo que q e r são proposições lógicas as quais possuem individualmente no máximo k átomos e, conjuntamente, possuem $k+1$ átomos:

$$q \wedge r \quad q \vee r \quad q \rightarrow r$$

Como, por hipótese de indução o valor-verdade de q e r é V, tem-se que, em qualquer dos casos, o valor-verdade de p é V. ❑

EXEMPLO 8.6 - Segundo Princípio da Indução Matemática: Postagem
Considere o seguinte teorema:

Qualquer valor de postagem igual ou maior que 12 reais
pode ser formado usando exclusivamente selos de 4 e de 5 reais

Uma prova por indução (no número de selos, usando a segunda versão do Segundo Princípio da Indução) é como segue:

a) *Base de Indução*. Seja $k \in \{12, 13, 14, 15\}$. Então, para cada k, vale:

12 reais pode ser formado com 3 selos de 4 reais;
13 reais pode ser formado com 2 selos de 4 reais e 1 selo de 5 reais;
14 reais pode ser formado com 1 selo de 4 reais e 2 selos de 5 reais;
15 reais pode ser formado com 3 selos de 5 reais;

b) *Hipótese de Indução*. Suponha que, para $k \in \mathbf{N}$, e para qualquer $u \in \mathbf{N}$ tal que $15 \leq u \leq k$, se o valor de postagem é u, então pode ser formado usando selos de 4 e de 5 reais;

c) *Passo de Indução*. Seja uma postagem cujo valor é $k+1$ reais. Então tal postagem pode ser formada usando uma postagem de $k-3$ reais mais um selo de 4 reais. ❑

8.4 Definição Indutiva

O Princípio da Indução Matemática pode ser usado também em definições. Uma definição de uma construção usando esse princípio é denominada *definição indutiva* ou *definição recursiva*. Nesse caso, afirma-se que a construção é *indutivamente definida* ou *recursivamente definida*. Resumidamente, em uma definição indutiva:

- a base de indução explicita os casos elementares (mais simples);
- o passo de indução determina como os demais casos são definidos em termos dos anteriores.

Esse tipo de definição já foi informalmente apresentado anteriormente, conforme ilustrado no seguinte exemplo.

EXEMPLO 8.7 - Definição Indutiva: Fecho Transitivo
Suponha uma endorrelação $R: A \rightarrow A$. A construção do *Fecho Transitivo* de R, denotado por R^+, é definida como segue:

a) Se $\langle a, b \rangle \in R$, então $\langle a, b \rangle \in R^+$
b) Se $\langle a, b \rangle \in R^+$ e $\langle b, c \rangle \in R^+$, então $\langle a, c \rangle \in R^+$
c) Os únicos elementos de R^+ são os construídos como acima. ❑

No EXEMPLO 8.7 - Definição Indutiva: Fecho Transitivo, pode-se afirmar que:

- item a) é a base de indução;
- item b) é o passo de indução (e a hipótese?);
- item c) garante que, de fato, é uma definição indutiva.

Quando se afirma que se trata de uma definição indutiva, o item c) é usualmente omitido (pois é subentendido).

No exemplo que segue, lembre-se de que, para um dado alfabeto Σ, o conjunto de todas as palavras possíveis sobre Σ é denotado por Σ^*. Por exemplo, para o alfabeto $\Sigma = \{a, b\}$:

$$\Sigma^* = \{\varepsilon, a, b, aa, ab, ba, bb, aaa, \ldots\}$$

EXEMPLO 8.8 - Definição Indutiva: Conjunto de Todas as Palavras
Para um alfabeto Σ qualquer, o conjunto Σ^* pode ser indutivamente definido como segue:

a) *Base de Indução*.

$\varepsilon \in \Sigma^*$
para qualquer $x \in \Sigma$, vale $x \in \Sigma^*$

b) *Passo de Indução*.

Se u e v são palavras de Σ^*,
então a concatenação uv é uma palavra de Σ^* ❑

O conceito de *Fórmula Lógica* ou simplesmente *Fórmula* (palavra da *Linguagem Lógica*) foi informalmente introduzido como sendo uma sentença lógica corretamente construída sobre o alfabeto cujos símbolos são conetivos ($\wedge$, $\vee$, $\rightarrow$, etc.), parênteses, identificadores (p, q, r, etc.), constantes, etc. No exemplo a seguir, é apresentada uma definição indutiva de fórmula.

EXEMPLO 8.9 - Definição Indutiva: Fórmula Lógica
Para facilitar o entendimento, a definição indutiva de fórmula lógica que segue é simplificada, não incluindo quantificadores, variáveis, etc.:

a) *Base de Indução*. Qualquer proposição atômica (incluindo V e F) é uma fórmula;

b) *Passo de Indução*. Se q e r são fórmulas, então também são fórmulas:

$(\neg q)$
$(q \wedge r)$
$(q \vee r)$
$(q \rightarrow r)$
$(q \leftrightarrow r)$ ❑

Como ilustração da relação entre o Princípio da Indução Matemática e a definição indutiva observe o seguinte exemplo e compare-o com o EXEMPLO 8.9 - Definição Indutiva: Fórmula Lógica.

EXEMPLO 8.10 - Definição Indutiva × Princípio da Indução Matemática

Uma definição indutiva de fórmula lógica, usando o Segundo Princípio da Indução, é como segue:

a) *Base de Indução*. Seja $k=0$. Então qualquer proposição atômica (incluindo os valores verdade V e F) é uma fórmula;

b) *Hipótese de Indução*. Suponha que, para algum $k \in \mathbf{N}$, e para qualquer $i \in \mathbf{N}$ tal que $i \leq k$, se p é uma fórmula com i conetivos, então p é uma fórmula lógica;

c) *Passo de Indução*. Seja p uma fórmula com $k+1$ conetivos. Então p pode ser reescrita em um dos seguintes casos, sendo que q e r são fórmulas e possuem conjuntamente k conetivos:

$(\neg q)$
$(q \wedge r)$
$(q \vee r)$
$(q \rightarrow r)$
$(q \leftrightarrow r)$ ❑

Como já destacado anteriormente, embora o Princípio da Indução Matemática seja definido sobre o conjunto dos números naturais, qualquer conjunto isomorfo a $\mathbf{N}$ pode ser considerado. Os exemplos que seguem ilustram definições indutivas sobre $\mathbf{N}^2$.

EXEMPLO 8.11 - Definição Indutiva sobre $\mathbf{N}^2$: Adição

Suponha que se deseja definir a adição nos naturais $ad: \mathbf{N}^2 \rightarrow \mathbf{N}$ usando-se exclusivamente a constante zero e a função sucessor $suc: \mathbf{N} \rightarrow \mathbf{N}$ a qual, para qualquer $n \in \mathbf{N}$, $suc\langle n \rangle = n+1$. Uma definição indutiva é a seguinte:

a) *Base de Indução*.

$ad\langle 0, 0 \rangle = 0$

b) *Passo de Indução*. Se a e b são número naturais, então:

$ad\langle a+1, 0 \rangle = suc\langle ad\langle a, 0 \rangle \rangle$
$ad\langle a, b+1 \rangle = suc\langle ad\langle a, b \rangle \rangle$

Como ilustração, a adição de 1 e 2 seria como segue:

$ad\langle 1, 2 \rangle =$
$suc\langle ad\langle 1, 1 \rangle \rangle =$
$suc\langle suc\langle ad\langle 1, 0 \rangle \rangle \rangle =$
$suc\langle suc\langle suc\langle ad\langle 0, 0 \rangle \rangle \rangle \rangle \rangle =$
$suc\langle suc\langle suc\langle 0 \rangle \rangle \rangle =$
$suc\langle suc\langle 1 \rangle \rangle =$
$suc\langle 2 \rangle = 3$ ❑

EXEMPLO 8.12 - Definição Indutiva sobre $\mathbf{N}^2$: Mínimo

Suponha que se deseja definir a função $min: \mathbf{N}^2 \rightarrow \mathbf{N}$ a qual determina o menor de dois números fornecidos. Por exemplo, $min\langle 2, 3 \rangle = 2$. Uma definição indutiva é a seguinte:

$min\langle a, 0 \rangle = 0$
$min\langle 0, b \rangle = 0$
$min\langle a+1, b+1 \rangle = min\langle a, b \rangle + 1$

Como ilustração, o cálculo de $min\langle 2, 3 \rangle$ é como segue:

$min\langle 2, 3 \rangle =$
$min\langle 1, 2 \rangle + 1 =$
$min\langle 0, 1 \rangle + 1 + 1 =$
$0 + 1 + 1 = 2$ ❑

8.5 Expressões Regulares

No Capítulo 6 - Funções Parciais e Totais, foi introduzido o formalismo Autômato Finito, o qual é especialmente importante, pois define a classe das *linguagens regulares* e possui diversas aplicações na Computação e Informática.

Linguagens regulares (prova-se) podem ser alternativamente definidas usando *Expressões Regulares*. A definição que segue usa o Segundo Princípio da Indução Matemática (segunda versão).

Definição 8.3 - Expressão Regular

Uma *Expressão Regular* sobre um alfabeto Σ é indutivamente definida como segue:

a) *Base de Indução*.

a.1) $\varnothing$ é uma expressão regular e denota a linguagem vazia;

a.2) ε é uma expressão regular e denota a linguagem contendo exclusivamente a palavra vazia, ou seja, $\{\varepsilon\}$;

a.3) Qualquer símbolo $x \in \Sigma$ é uma expressão regular e denota a linguagem contendo a palavra unitária x, ou seja, $\{x\}$;

b) *Passo de Indução*. Se r e s são expressões regulares e denotam as linguagens R e S, respectivamente, então:

b.1) *União*. $(r+s)$ é expressão regular e denota a linguagem:

$$R \cup S$$

b.2) *Concatenação*. (rs) é expressão regular e denota a linguagem:

$$RS = \{uv \mid u \in R \text{ e } v \in S\}$$

b.3) *Concatenação Sucessiva*. (r^*) é expressão regular e denota a linguagem:

$$R^*$$ ❑

EXEMPLO 8.13 - Expressão Regular

Na tabela da Figura 8.2, são apresentadas expressões regulares e as correspondentes linguagens. Observe que a omissão de parênteses em uma expressão regular é usual, respeitando-se as seguintes convenções:

- a concatenação sucessiva tem precedência sobre a concatenação e a união;
- a concatenação tem precedência sobre a união.

❑

Expressão Regular	Linguagem Representada
aa	somente a palavra aa
ba*	todas as palavras que iniciam por b, seguido por zero ou mais a
(a + b)*	todas as palavras sobre {a, b}
(a + b)*aa(a + b)*	todas as palavras contendo aa como subpalavra
a*ba*ba*	todas as palavras contendo exatamente dois b
(a + b)*(aa + bb)	todas as palavras que terminam com aa ou bb
(a + ε)(b + ba)*	todas as palavras que não possuem dois a consecutivos

Figura 8.2 Expressões Regulares e as correspondentes linguagens representadas

8.6 Computações de um Autômato Finito

Alguns modelos computacionais já foram introduzidos como, por exemplo:

- Rede de Petri, definida como uma relação;
- Autômato Finito, definido como uma função parcial.

Ambos os modelos foram formalmente definidos *sintaticamente*. Ou seja, para os dois modelos, foram apresentadas as definições formais de sua *forma*. Já a *semântica*, ou seja, uma interpretação do funcionamento ou processamento de cada modelo, foi apresentada textualmente, de maneira informal.

Quando da apresentação do modelo Autômato Finito, foi afirmado que:

o processamento de um Autômato Finito
é a sucessiva aplicação de suas computações atômicas (transições)

Para facilitar o entendimento do que segue, é reproduzida na Figura 8.3 a sintaxe de um Autômato Finito (esquerda) na forma de diagrama e uma ilustração de seu processamento (direita), para a entrada w = abba, a partir do estado inicial q_0.

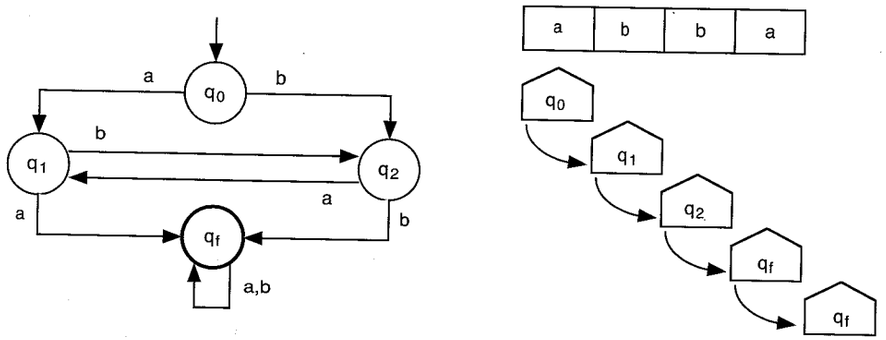


Figura 8.3 Autômato Finito (esquerda) e correspondente seqüência de processamento (direita)

Supondo um conjunto finito de estados Q e um alfabeto Σ, lembre-se de que o programa de um Autômato Finito pode ser definido como uma função parcial:

$$\delta: Q \times \Sigma \rightarrow Q$$

na qual, um par da forma ⟨⟨q, a⟩, p⟩, ou seja, tal que δ⟨q, a⟩ = p, indica que, no estado q, ao ler o símbolo a, o Autômato assume o estado p.

Assim, para definir formalmente o comportamento de um Autômato Finito (ou seja, dar semântica à sintaxe de um Autômato Finito) para uma dada palavra, é necessário *estender* a definição da função programa usando como argumento um estado e uma *palavra*. A função programa estendida a uma palavra, que determina as *computações* de um autômato finito a partir de um dado estado, pode ser definida indutivamente como segue:

- *base de indução*: ao processar a entrada vazia (palavra vazia), o Autômato permanece no mesmo estado;
- *passo de indução*: ao processar uma entrada não-vazia (palavra não-vazia), o Autômato processa o primeiro símbolo da entrada, usando a função programa δ e aplica o mesmo raciocínio sucessivamente ao resto da palavra.

Seja $\delta: Q \times \Sigma \rightarrow Q$ a função programa de um Autômato Finito. Então a *função programa estendida*, denotada por:

$$\delta^*: Q \times \Sigma^* \rightarrow Q$$

é a função programa $\delta: Q \times \Sigma \rightarrow Q$ estendida para palavras e é indutivamente definida como segue (suponha a um símbolo qualquer de Σ e u = aw uma palavra qualquer sobre Σ* cujo primeiro símbolo é a e o resto é uma subpalavra w):

$$\delta^*\langle q, \varepsilon\rangle = q$$
$$\delta^*\langle q, aw\rangle = \delta^*\langle\delta\langle q, a\rangle, w\rangle$$

EXEMPLO 8.14 - Função Programa Estendida

Considere o Autômato Finito ilustrado na Figura 8.3. Então a função programa estendida aplicada à palavra abba a partir do estado inicial q_0 é como segue:

$\delta^*\langle q_0, abba\rangle =$	função estendida sobre abba
$\delta^*\langle\delta\langle q_0, a\rangle, bba\rangle =$	processa *a*bba
$\delta^*\langle q_1, bba\rangle =$	função estendida sobre bba
$\delta^*\langle\delta\langle q_1, b\rangle, ba\rangle =$	processa *b*ba
$\delta^*\langle q_2, ba\rangle =$	função estendida sobre ba
$\delta^*\langle\delta\langle q_2, b\rangle, a\rangle =$	processa *b*a
$\delta^*\langle q_f, a\rangle =$	função estendida sobre a
$\delta^*\langle\delta\langle q_f, a\rangle, \varepsilon\rangle =$	processa *a*
$\delta^*\langle q_f, \varepsilon\rangle = q_f$	função estendida sobre ε: fim da indução

Portanto, a palavra abba é aceita. ❑

8.7 Leitura Complementar: Gramática e BNF

As *Gramáticas de Chomsky* constituem um formalismo universalmente aceito e muito usado em Computação e Informática e em outras áreas como, por exemplo:

- *Linguagens Formais*: definição e estudo das propriedades das linguagens em geral;
- *Teoria da Computação*: estudo dos limites da solucionabilidade de problemas e suas propriedades;
- *Linguagens Naturais*: estudo de linguagens como o português;
- *Sistemas Biológicos*: simulação do desenvolvimento de sistemas vivos.

Prova-se que o conceito de gramática é equivalente ao de *Máquina de Turing* no que se refere ao poder computacional. Portanto, segundo a *Hipótese de Church*, qualquer *função computável* pode ser especificada por uma gramática. Logo, o conceito de gramática pode ser considerado como uma definição formal de *algoritmo* (embora a noção de algoritmo seja intuitiva, razão pela qual a Hipótese de Church não pode ser demonstrada).

Em Computação e Informática, gramáticas são especialmente importantes para definir a sintaxe de linguagens de programação. Em particular, a *sintaxe* de linguagens como Pascal, C, etc., é definida usando gramáticas.

8.7.1 Gramática

Uma gramática é, basicamente, um conjunto finito de regras as quais, quando aplicadas sucessivamente, geram palavras. A geração de uma palavra é indutivamente definida. O conjunto de todas as palavras geradas define a linguagem. Assim, trata-se de uma forma finita de expressar a sintaxe de linguagens eventualmente infinitas. As gramáticas usadas para as linguagens naturais como Português são as mesmas que as usadas para linguagens artificiais como Pascal. Eventualmente, gramáticas também são usadas para definir *semântica*. Entretanto, para tratar semântica, em geral, são usados outros tipos de formalismos.

Definição 8.4 - Gramática

Uma *Gramática de Chomsky* ou simplesmente *Gramática*, é composta por:

a) V, um conjunto finito de símbolos *variáveis* ou *não-terminais*;
b) T, um conjunto finito de símbolos *terminais* disjunto de V;
c) $P: (V \cup T)^+ \rightarrow (V \cup T)^*$, uma relação finita (ou seja, um conjunto finito de pares). Cada par da relação é denominado de *regra de produção*;
d) S, um elemento distinguido de V denominado *símbolo inicial* ou *variável inicial*. ❑

Uma regra de produção $\langle \alpha, \beta \rangle$ é representada como segue:

$$\alpha \rightarrow \beta$$

Por simplicidade, um grupo de regras de produção da forma $\alpha \rightarrow \beta_1$, $\alpha \rightarrow \beta_2$, ..., $\alpha \rightarrow \beta_n$ (mesma componente no lado esquerdo) é usualmente abreviado como uma única produção na forma:

$$\alpha \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_n$$

As regras de produção definem as condições de geração das palavras da linguagem. A aplicação de uma regra de produção é denominada *derivação* de uma palavra e é formalmente definida como um par de uma relação. A aplicação sucessiva de regras de produção (fecho transitivo da relação de derivação) permite derivar as palavras da linguagem representada pela gramática.

Definição 8.5 - Relação de Derivação

Seja G uma gramática tal que $P: (V \cup T)^+ \rightarrow (V \cup T)^*$. Uma *Derivação* é um par da *Relação de Derivação* denotada por $\Rightarrow$ com domínio em $(V \cup T)^+$ e contradomínio em $(V \cup T)^*$. Um par $\langle \alpha, \beta \rangle$ da relação de derivação é representado de forma infixada como segue:

$$\alpha \Rightarrow \beta$$

A relação de derivação $\Rightarrow$ é indutivamente definida como segue:

a) Para toda produção da forma $S \rightarrow \beta$ (S é o símbolo inicial de G), o seguinte par pertence à relação de derivação:

$$S \Rightarrow \beta$$

b) Para todo par $\eta \Rightarrow \rho\, \alpha\, \sigma$ da relação de derivação, se $\alpha \rightarrow \beta$ é regra de P, então o seguinte par também pertence à relação de derivação:

$$\eta \Rightarrow \rho\, \beta\, \sigma$$ ❑

Portanto, uma derivação é a substituição de uma subpalavra de acordo com uma regra de produção. A partir de uma gramática, é possível derivar ou gerar todas as palavras da linguagem que ela representa.

Definição 8.6 - Linguagem Gerada (por uma Gramática)

Seja G uma gramática. A *Linguagem Gerada* pela gramática G, denotada por $\text{Linguagem}(G)$, é composta por todas as palavras de símbolos terminais deriváveis a partir do símbolo inicial S. Assim, supondo que $\Rightarrow^+$ denota o fecho transitivo da relação de derivação $\Rightarrow$:

$$\text{Linguagem}(G) = \{ w \in T^* \mid S \Rightarrow^+ w \}$$ ❑

Portanto, uma palavra de uma linguagem não pode conter variáveis.

EXEMPLO 8.15 - Gramática e Derivação: Números Naturais

Suponha que se deseja definir uma gramática capaz de gerar qualquer número natural válido em uma linguagem de programação. Assim, a gramática G na qual:

$V = \{N, D\}$, sendo que N é o símbolo inicial
$T = \{0, 1, 2, \ldots, 9\}$
P é constituído pelas seguintes regras:

$N \rightarrow D$
$N \rightarrow DN$
$D \rightarrow 0 \mid 1 \mid \ldots \mid 9$

gera, sintaticamente, o conjunto dos números naturais. Note-se que se distinguem os zeros à esquerda. Por exemplo, distingue-se 123 de 0123 (sugere-se como exercício o desenvolvimento de uma gramática a qual não distingue zeros à esquerda). Como ilustração, uma derivação do número 243 é como segue (na coluna à direita, é apresentada a regra usada em cada passo de derivação):

$N \Rightarrow$	$N \rightarrow DN$
$DN \Rightarrow$	$D \rightarrow 2$
$2N \Rightarrow$	$N \rightarrow DN$
$2DN \Rightarrow$	$D \rightarrow 4$
$24N \Rightarrow$	$N \rightarrow D$
$24D \Rightarrow$	$D \rightarrow 3$
243	

Existe mais alguma derivação do número 243? ❑

Observe que, no exemplo acima, a seguinte interpretação indutiva pode ser dada à gramática em questão:

- *Base de Indução*: todo dígito é um número natural;
- *Passo de Indução*: se n é um número natural, então a concatenação de n com qualquer dígito também é um número natural.

EXEMPLO 8.16 - Gramática e Derivação: Expressão Aritmética

A linguagem gerada pela gramática G abaixo é composta por expressões aritméticas contendo parênteses balanceados, dois operadores e um operando:

$V = \{E\}$, sendo que E é o símbolo inicial

$T = \{+, *, (,), x\}$

P é constituído pelas seguintes regras (qual interpretação indutiva pode ser dada a este conjunto de regras?):

$E \rightarrow E+E$

$E \rightarrow E*E$

$E \rightarrow (E)$

$E \rightarrow x$

Por exemplo, a expressão aritmética $(x+x)*x$ pode ser gerada pela seguinte seqüência de derivação (identifique, claramente, em cada derivação, qual a regra de produção usada):

$$E \Rightarrow E*E \Rightarrow (E)*E \Rightarrow (E+E)*E \Rightarrow (x+E)*E \Rightarrow (x+x)*E \Rightarrow (x+x)*x$$

É possível gerar a mesma expressão com outras seqüências de derivação? ❑

8.7.2 BNF

Em Computação e Informática, uma forma usual de representar uma gramática é usando a *Forma de Backus Naur* ou simplesmente *BNF* (do inglês, *Backus Naur Form*). Em uma BNF, tem-se que:

- as variáveis são palavras delimitas pelos símbolos ⟨ e ⟩
- as palavras não-delimitadas são terminais;
- uma regra de produção $\langle \alpha, \beta \rangle$ é representada por:

$$\alpha ::= \beta$$

EXEMPLO 8.17 - BNF: Identificador em Pascal

Suponha que se deseja definir uma BNF capaz de gerar qualquer identificador válido na linguagem de programação Pascal, ou seja, na qual:

- *Base de Indução*: toda letra é um identificador;
- *Passo de Indução*: se S é um identificador, então a concatenação de S com qualquer letra ou dígito também é um identificador.

Uma BNF é como segue (a variável ⟨identificador⟩ é o símbolo inicial)

⟨identificador⟩ ::= ⟨letra⟩ | ⟨identificador⟩⟨letra⟩ | ⟨identificador⟩⟨dígito⟩

⟨letra⟩ ::= a | b | ... | z

⟨dígito⟩ ::= 0 | 1 | ... | 9 ❑

8.8 Recursão

Um conceito próximo ao de indução e presente na grande maioria das linguagens de programação é o de *recursão*. De fato, o conceito de recursão é inspirado no formalismo *Funções Recursivas de Kleene*, o qual é equivalente ao da *Máquina de Turing* e ao da *Gramática de Chomsky* no que se refere ao poder computacional. Portanto, segundo a *Hipótese de Church*, qualquer *função computável* pode ser especificada por uma função recursiva. Ou seja, qualquer *algoritmo* pode ser expresso usando recursividade.

De fato, toda definição indutiva pode ser simulada por uma *recursão* (se a linguagem de programação possui essa facilidade). Entretanto, nem toda recursão possui uma correspondente definição indutiva, pois não necessariamente a recursão respeita a boa ordem da indução. Destaca-se que, na maioria das aplicações computacionais, é recomendável programar uma recursão respeitando os princípios da definição indutiva pois, nesse caso, é garantido (pela boa ordem) que o programa pára e atinge o resultado esperado.

O exemplo que segue é especialmente importante para o entendimento da recursão de uma linguagem de programação.

EXEMPLO 8.18 - Definição Indutiva: Fatorial

Para um dado número natural n, define-se o fatorial n! como segue:

$n! = 1$, se $n = 0$

$n! = n*(n-1)*(n-2)\ldots*1$, se $n > 0$

Para o caso $n > 0$, observe que n! pode ser reescrito como segue:

$$n*(n-1)! \quad \text{sendo que} \quad (n-1)! = (n-1)*(n-2)\ldots*1$$

Da mesma forma, (n - 1)! pode ser reescrito como segue:

$$(n-1)*(n-2)! \quad \text{sendo que} \quad (n-2)! = (n-2)*(n-3)*\ldots*1$$

e assim sucessivamente. Portanto, o fatorial de um número n pode ser determinado multiplicando n pelo fatorial de seu antecessor n - 1. Tal raciocínio pode ser recursivamente aplicado até chegar ao fatorial de zero. Logo, a função fatorial pode ser definida em termos dela mesma, até atingir o fatorial de zero, como segue:

a) *Base de Indução.*

$0! = 1$

b) *Passo de Indução.*

$n! = n*(n-1)!$

Exemplificando, o cálculo do fatorial de 4 é como segue:

$4! = 4*(4-1)! = 4*3! =$ passo de indução

$4*3*(3-1)! = 4*3*2! =$ passo de indução

$4*3*2*(2-1)! = 4*3*2*1! =$ passo de indução

$4*3*2*1*(1-1)! = 4*3*2*1*0! =$ base de indução

$4*3*2*1*1 = 24$ ❑

O mesmo princípiopode ser adotado nas linguagens de programação, usando o conceito de *recursão*, ou seja, o de uma função definida em termos dela mesma. Essa definição pode ser direta (uma função referencia a si mesma) ou indireta (uma função referencia outra função que, por sua vez, direta ou indiretamente, referencia a primeira).

EXEMPLO 8.19 - Função Recursiva em Pascal: Fatorial

A seguinte função em Pascal implementa o fatorial usando recursão. Compare com a definição indutiva apresentada no EXEMPLO 8.18 - Definição Indutiva: Fatorial:

```
function fatorial(n: integer): integer;
    begin
    if    n = 0
    then  fatorial := 1
    else  fatorial := n * fatorial(n - 1)
    end
```

O cálculo da função fatorial em Pascal é análogo ao do EXEMPLO 8.18 - Definição Indutiva: Fatorial. ❑

Como ilustração, segue a mesma função em Haskell.

EXEMPLO 8.20 - Função recursiva em Haskell: Fatorial

A seguinte função em Haskell implementa o fatorial usando recursão. Observe que é praticamente a mesma definição apresentada no EXEMPLO 8.18 - Definição Indutiva: Fatorial:

```
fatorial(0) = 1
fatorial(n) = n * fatorial(n - 1)
```

❑

Como já afirmado, nem toda programação recursiva corresponde a uma definição indutiva, como ilustrado no exemplo a seguir.

EXEMPLO 8.21 - Função Recursiva

A seguinte função em Pascal implementa uma função a qual, quando chamada com qualquer parâmetro atual, fica processando indefinidamente (*loop* infinito).

```
function ciclo_infinito(x: real): real;
    begin
    ciclo_infinito := x * ciclo_infinito(x)
    end
```

Observe que a função definida não corresponde a qualquer noção de boa ordem e, portanto, não caracteriza uma definição indutiva. ❑

Observação 8.7 - Eficiência da Recursão

Em *linguagens imperativas*, tais como C e Pascal, toda função recursiva segue um padrão de processamento que, em geral, resulta em uma considerável demanda do recurso memória. Por isso, alguns textos didáticos sobre linguagens imperativas reconhecem o papel da recursão como uma construção que permite expressar certos programas de forma mais elegante, mas não a recomendam pois consideram que um programa recursivo sempre pode ser escrito iterativamente de forma mais eficiente. O padrão para execução de um comando sucessivamente é a iteração baseada em comandos de blocos (tais como os comandos `while` ou `for`).

No caso de *linguagens funcionais*, como Haskell, recursão é a técnica padrão de aplicar uma operação diversas vezes. Como recursão é base da maioria dos programas funcionais, os compiladores e interpretadores, em geral, executam otimizações que não são padrão em linguagens imperativas, tais como o reconhecimento de recursão de cauda que pode ser implementada de forma tão eficiente quanto iteração. O estudo de tais técnicas não é detalhado neste livro. ❑

8.9 Leitura Complementar: Funções Recursivas Parciais

As *Funções Recursivas Parciais* ou *Funções Recursivas de Kleene*, introduzidas por S. C. Kleene (1936), como o próprio nome indica, são funções parciais definidas recursivamente.

Assim como Turing, Kleene tinha como objetivo formalizar a noção intuitiva de função computável e, conseqüentemente, do que é possível computar em um computador. Quando foi provado que a Classe das Funções Turing-Computáveis era igual à Classe das Funções Recursivas Parciais, a Hipótese de Church cresceu significativamente em termos de credibilidade. A Hipótese de Church, apresentada no Capítulo 2 - Lógica e Técnicas de Demonstração, afirma que:

qualquer função computável pode ser processada por uma Máquina de Turing, ou seja, existe um procedimento expresso na forma de uma Máquina de Turing capaz de processar a função

Relativamente às Funções Recursivas Parciais, verifica-se que a composição de três funções naturais simples:

- constante zero;
- sucessor;
- projeção;

juntamente com as seguintes operações:

- substituição;
- recursão primitiva;
- minimização;

constitui uma forma compacta e natural para definir muitas funções e suficientemente poderosa para descrever toda função intuitivamente computável.

8.9.1 Substituição

A substituição definida a seguir generaliza o conceito usual de composição de funções.

Definição 8.8 - Substituição

Sejam $g: \mathbf{N}^k \rightarrow \mathbf{N}$ e $f_1, f_2, \ldots, f_k: \mathbf{N}^n \rightarrow \mathbf{N}$ funções parciais. A função parcial:

$$h: \mathbf{N}^n \rightarrow \mathbf{N}$$

é a *Substituição de Funções* definida a partir de $g, f_1, f_2, \ldots, f_k$ como segue:

$$h\langle x_1, x_2, \ldots, x_n\rangle = g\langle f_1\langle x_1, x_2, \ldots, x_n\rangle, f_2\langle x_1, x_2, \ldots, x_n\rangle, \ldots, f_k\langle x_1, x_2, \ldots, x_n\rangle\rangle$$

A função parcial h é dita *definida* para $\langle x_1, x_2, \ldots, x_n\rangle$ se, e somente se:

$f_i\langle x_1, x_2, \ldots, x_n\rangle$ é definida para todo $i \in \{1, 2, \ldots, k\}$

$g\langle f_1\langle x_1, x_2, \ldots, x_n\rangle, f_2\langle x_1, x_2, \ldots, x_n\rangle, \ldots, f_k\langle x_1, x_2, \ldots, x_n\rangle\rangle$ é definida. ❑

EXEMPLO 8.22 - Substituição de Funções: Funções Constantes

Suponha as seguintes funções:

$const_{zero}: \mathbf{N} \rightarrow \mathbf{N}$ constante zero: $const_{zero}\langle x\rangle = 0$

$suc: \mathbf{N} \rightarrow \mathbf{N}$ sucessor: $suc\langle x\rangle = n + 1$

$ad: \mathbf{N}^2 \rightarrow \mathbf{N}$ adição: $ad\langle x, y\rangle = x + y$

As seguintes funções são definidas, usando substituição de funções:

$const_{um} = suc\langle const_{zero}\rangle : \mathbf{N} \to \mathbf{N}$ constante um: $const_{um}\langle x\rangle = 1$
$const_{dois} = suc\langle const_{um}\rangle : \mathbf{N} \to \mathbf{N}$ constante dois: $const_{dois}\langle x\rangle = 2$
$const_{três} = ad\langle const_{um}, const_{dois}\rangle : \mathbf{N} \to \mathbf{N}$ constante três: $const_{três}\langle x\rangle = 3$

Sugere-se, como exercício, verificar se, usando as mesmas funções $const_{zero}$, suc e ad, existem outras formas de definir equivalentemente as funções $const_{um}$, $const_{dois}$ e $const_{três}$. ❑

8.9.2 Recursão Primitiva

Definição 8.9 - Recursão Primitiva
Sejam $f: \mathbf{N}^n \to \mathbf{N}$ e $g: \mathbf{N}^{n+2} \to \mathbf{N}$ funções parciais. A função parcial:

$$h: \mathbf{N}^{n+1} \to \mathbf{N}$$

é definida por *Recursão Primitiva* a partir de f e g como segue:

$$h\langle x_1, x_2, \ldots, x_n, 0\rangle = f\langle x_1, x_2, \ldots, x_n\rangle$$
$$h\langle x_1, x_2, \ldots, x_n, y+1\rangle = g\langle x_1, x_2, \ldots, x_n, y, h\langle x_1, x_2, \ldots, x_n, y\rangle\rangle$$

A função parcial h é dita definida para $\langle x_1, x_2, \ldots, x_n, y\rangle$ se e somente se:

$f\langle x_1, x_2, \ldots, x_n\rangle$ é definida;
$g\langle x_1, x_2, \ldots, x_n, i, h\langle x_1, x_2, \ldots, x_n, i\rangle\rangle$ é definida para todo $i \in \{1, 2, \ldots, y\}$ ❑

EXEMPLO 8.23 - Recursão Primitiva: Adição
Suponha as seguintes funções:

$id: \mathbf{N} \to \mathbf{N}$ identidade: $id\langle x\rangle = x$
$suc: \mathbf{N} \to \mathbf{N}$ sucessor: $suc\langle x\rangle = x + 1$
$\pi_3: \mathbf{N}^3 \to \mathbf{N}$ projeção (3ª componente): $\pi_3\langle x, y, z\rangle = z$

A função adição nos naturais, tal que:

$ad: \mathbf{N}^2 \to \mathbf{N}$ adição: $ad\langle x, y\rangle = x + y$

é definida, usando recursão primitiva, como segue:

$$ad\langle x, 0\rangle = id\langle x\rangle$$
$$ad\langle x, y+1\rangle = \pi_3\langle x, y, suc\langle ad\langle x, y\rangle\rangle\rangle$$

Por exemplo, $ad\langle 3, 2\rangle$ é como abaixo. Note-se que, nas instâncias de $\pi_3\langle x, y, suc\langle ad\langle x, y\rangle\rangle\rangle$, somente a componente $suc\langle ad\langle x, y\rangle\rangle$ é importante na lógica apresentada. A função π_3, bem como as componentes x e y, estão presentes somente para satisfazer a definição de recursão primitiva. De fato, funções do tipo projeção são fundamentais em recursão primitiva, como será visto adiante.

$$\begin{aligned}
& ad\langle 3, 2\rangle = \\
& = \pi_3\langle 3, 1, suc\langle ad\langle 3, 1\rangle\rangle\rangle = \\
& = \pi_3\langle 3, 1, suc\langle \pi_3\langle 3, 1, suc\langle ad\langle 3, 0\rangle\rangle\rangle\rangle\rangle = \\
& = \pi_3\langle 3, 1, suc\langle \pi_3\langle 3, 1, suc\langle id\langle 3\rangle\rangle\rangle\rangle\rangle = \\
& = \pi_3\langle 3, 1, suc\langle \pi_3\langle 3, 1, suc\langle 3\rangle\rangle\rangle\rangle = \\
& = \pi_3\langle 3, 1, suc\langle \pi_3\langle 3, 1, 4\rangle\rangle\rangle = \\
& = \pi_3\langle 3, 1, suc\langle 4\rangle\rangle = \\
& = \pi_3\langle 3, 1, 5\rangle = \\
& = 5
\end{aligned}$$

❑

8.9.3 Minimização

O conceito de minimização que segue não é intuitivo na noção de funções recursivas parciais. Entretanto, é fundamental para garantir que a Classe de Funções Recursivas Parciais definida adiante possa conter qualquer função computável.

Definição 8.10 - Minimização
Seja $f: \mathbf{N}^{n+1} \to \mathbf{N}$ uma função parcial. A função parcial:

$$h: \mathbf{N}^n \to \mathbf{N}$$

é dita definida por *Minimização* de f e é tal que:

$$h\langle x_1, x_2, \ldots, x_n\rangle = \min\{y \mid f\langle x_1, \ldots, x_n, y\rangle = 0 \text{ e, } \forall z \text{ tal que } z < y, f\langle x_1, \ldots, x_n, z\rangle \text{ é definida}\}$$

❑

Portanto, a função h, para o valor $\langle x_1, \ldots, x_n\rangle$, é definida como o menor natural y tal que $f\langle x_1, \ldots, x_n, y\rangle = 0$. Adicionalmente, a condição:

$$\forall z \text{ tal que } z < y, f\langle x_1, \ldots, x_n, z\rangle \text{ é definida}$$

garante que é possível determinar, em um tempo finito, se, para qualquer valor z menor do que y, $f\langle x_1, \ldots, x_n, z\rangle$ é diferente de zero. Note que a função h é parcial (quais as condições para estar definida em $\langle x_1, \ldots, x_n\rangle$?).

Por simplicidade, no texto que segue, para uma função h definida por minimização de f, a seguinte notação é adotada (compare com a definição acima):

$$h\langle x_1, x_2, \ldots, x_n\rangle = \min\{y \mid f\langle x_1, \ldots, x_n, y\rangle = 0\}$$

EXEMPLO 8.24 - Minimização, Recursão: Número Zero
Suponha a função constante $const_{zero}: \mathbf{N} \to \mathbf{N}$ (sendo que $const_{zero}\langle x\rangle = 0$). Considere a seguinte função que identifica o número zero nos naturais, onde $\mathbf{N}^0 = 1$, sendo $1 = \{*\}$ um conjunto unitário fixo (compare com a função constante $const_{zero}$):

$zero: 1 \to \mathbf{N}$ número zero: $zero\langle * \rangle = 0$

A função número zero pode ser definida, usando minimização, como segue:

$$zero = \min\{y \mid const_{zero}\langle y\rangle = 0\}$$

De fato, o menor natural y tal que $const_{zero}\langle y\rangle = 0$ é 0. ❑

EXEMPLO 8.25 - Minimização, Recursão: Antecessor
Suponha a função número zero, bem como a seguinte função de projeção:

$\pi_1: \mathbf{N}^2 \to \mathbf{N}$ projeção (1ª componente): $\pi_1\langle x, y\rangle = x$

A seguinte função antecessor nos naturais:

$ant: \mathbf{N} \to \mathbf{N}$ antecessor

pode ser definida, usando recursão primitiva (supondo que antecessor de 0 é 0), como segue:

$$ant\langle 0\rangle = zero$$
$$ant\langle y+1\rangle = \pi_1\langle y, ant\langle y\rangle\rangle$$

Por exemplo, $ant\langle 2\rangle$ é como segue:

$$
\begin{aligned}
&\text{ant}\langle 2\rangle = \\
&= \pi_1\langle 1, \text{ant}\langle 1\rangle\rangle = \\
&= \pi_1\langle 1, \pi_1\langle 0, \text{ant}\langle 0\rangle\rangle\rangle = \\
&= \pi_1\langle 1, \pi_1\langle 0, \text{zero}\rangle\rangle = \\
&= \pi_1\langle 1, \pi_1\langle 0, 0\rangle\rangle = \\
&= \pi_1\langle 1, 0\rangle = \\
&= 1
\end{aligned}
$$

Repare que foi necessário usar uma função de projeção. ❑

EXEMPLO 8.26 - Minimização, Recursão: Subtração

Suponha a função número zero, bem como as seguintes funções:

$\text{id}: \mathbf{N} \to \mathbf{N}$ identidade: $\text{id}\langle x\rangle = x$

$\pi_3: \mathbf{N}^3 \to \mathbf{N}$ projeção (3ª componente): $\pi_3\langle x, y, z\rangle = z$

A seguinte função subtração nos naturais:

$\text{sub}: \mathbf{N}^2 \to \mathbf{N}$ subtração

pode ser definida usando recursão primitiva:

$$
\begin{aligned}
&\text{sub}\langle x, 0\rangle = \text{id}\langle x\rangle \\
&\text{sub}\langle x, y+1\rangle = \text{ant} \circ \pi_3\langle x, y, \text{sub}\langle x, y\rangle\rangle
\end{aligned}
$$

Por exemplo, $\text{sub}\langle 3, 2\rangle$ é como segue:

$$
\begin{aligned}
&\text{sub}\langle 3, 2\rangle = \\
&= \text{ant} \circ \pi_3\langle 3, 1, \text{sub}\langle 3, 1\rangle\rangle = \\
&= \text{ant} \circ \pi_3\langle 3, 1, \text{ant} \circ \pi_3\langle 3, 0, \text{sub}\langle 3, 0\rangle\rangle\rangle = \\
&= \text{ant} \circ \pi_3\langle 3, 1, \text{ant} \circ \pi_3\langle 3, 0, \text{id}\langle 3\rangle\rangle\rangle = \\
&= \text{ant} \circ \pi_3\langle 3, 1, \text{ant} \circ \pi_3\langle 3, 0, 3\rangle\rangle = \\
&= \text{ant} \circ \pi_3\langle 3, 1, \text{ant}\langle 3\rangle\rangle = \\
&= \text{ant} \circ \pi_3\langle 3, 1, 2\rangle = \\
&= \text{ant}\langle 2\rangle = \\
&= 1
\end{aligned}
$$

Novamente, foi necessário usar uma função de projeção. Sugere-se como exercício determinar o valor de $\text{sub}\langle 2, 3\rangle$. ❑

8.9.4 Função Recursiva Parcial

Funções recursivas parciais são definidas a partir de três funções básicas sobre o conjunto dos números naturais, como segue:

- zero;
- sucessor;
- projeção.

Mais precisamente, projeção não é uma função, mas uma família de funções, pois depende do número de componentes, bem como de qual componente deseja-se projetar. É interessante observar que, somente com essas três funções, bem como com as construções de substituição, recursão primitiva e minimização, é possível definir qualquer função intuitivamente computável.

Definição 8.11 - Função Recursiva Parcial

Uma *Função Recursiva Parcial* ou *Função Recursiva de Kleene* é indutivamente definida como segue:

a) *Funções Básicas*. As seguintes funções são recursivas parciais:

$\text{zero}: 1 \to \mathbf{N}$ constante zero: $\text{zero}\langle * \rangle = 0$

$\text{suc}: \mathbf{N} \to \mathbf{N}$ sucessor: $\text{suc}\langle x\rangle = x + 1$

$\pi_i: \mathbf{N}^n \to \mathbf{N}$ projeção (i-ésima componente): $\pi_i\langle x_1, x_2, \ldots, x_n\rangle = x_i$

b) *Substituição de Funções*. Se $g: \mathbf{N}^k \to \mathbf{N}$ e $f_1, f_2, \ldots, f_k: \mathbf{N}^n \to \mathbf{N}$ são funções recursivas parciais, então a seguinte função é recursiva parcial:

$$h: \mathbf{N}^n \to \mathbf{N}$$

e é definida pela *substituição de funções* a partir de $g, f_1, f_2, \ldots, f_k$ como segue:

$$h\langle x_1, x_2, \ldots, x_n\rangle = g\langle f_1\langle x_1, x_2, \ldots, x_n\rangle, f_2\langle x_1, x_2, \ldots, x_n\rangle, \ldots, f_k\langle x_1, x_2, \ldots, x_n\rangle\rangle$$

c) *Recursão Primitiva*. Se $f: \mathbf{N}^n \to \mathbf{N}$ e $g: \mathbf{N}^{n+2} \to \mathbf{N}$ são funções recursivas parciais, então a seguinte função é recursiva parcial:

$$h: \mathbf{N}^{n+1} \to \mathbf{N}$$

e é definida por *recursão primitiva* a partir de f e g como segue:

$$
\begin{aligned}
&h\langle x_1, x_2, \ldots, x_n, 0\rangle = f\langle x_1, x_2, \ldots, x_n\rangle \\
&h\langle x_1, x_2, \ldots, x_n, y+1\rangle = g\langle x_1, x_2, \ldots, x_n, y, h\langle x_1, x_2, \ldots, x_n, y\rangle\rangle
\end{aligned}
$$

d) *Minimização*. Se $f: \mathbf{N}^{n+1} \to \mathbf{N}$ é função recursiva parcial, então a seguinte função é recursiva parcial:

$$h\ \mathbf{N}^n \to \mathbf{N}$$

e é definida por *minimização* de f, como segue:

$$h\langle x_1, x_2, \ldots, x_n\rangle = \min\{\, y \mid f\langle x_1, \ldots, x_n, y\rangle = 0 \,\}$$ ❑

EXEMPLO 8.27 - Funções Recursivas Parciais

a) *Função Identidade*. A função identidade:

$\text{id}: \mathbf{N} \to \mathbf{N}$ identidade: $\text{id}\langle x\rangle = x$

é recursiva parcial, pois é uma função básica de projeção, ou seja:

$$\text{id} = \pi_1$$

b) As seguintes funções, exemplificadas anteriormente, são recursivas parciais:

$\text{ad}: \mathbf{N}^2 \to \mathbf{N}$ adição

$\text{sub}: \mathbf{N}^2 \to \mathbf{N}$ subtração

$\text{const}_{\text{um}}: \mathbf{N} \to \mathbf{N}$ constante um

$\text{const}_{\text{dois}}: \mathbf{N} \to \mathbf{N}$ constante dois

$\text{const}_{\text{três}}: \mathbf{N} \to \mathbf{N}$ constante três

$\text{zero}: \{*\} \to \mathbf{N}$ número zero

$\text{ant}: \mathbf{N} \to \mathbf{N}$ antecessor

❑

O seguinte teorema não será demonstrado.

Teorema 8.12 - Funções Recursivas Parciais × Máquinas de Turing

A classe das Funções Recursivas Parciais é equivalente à classe das Funções Turing-Computáveis ❑

8.10 Exercícios

Exercício 8.1 Prove por indução que, para qualquer $n \in \mathbf{N}$:

a) $2+4+6+\ldots+2n = n(n+1)$
b) $1+3+6+\ldots+n(n+1)/2 = n(n+1)(n+2)/6$
c) $1+8+27\ldots+n^3 = (1+2+\ldots+n)^2$
d) $1^2+3^2+5^2\ldots+(2n-1)^2 = n(2n-1)(2n+1)/3$
e) $1*3+2*4+3*5\ldots+n(n+2) = n(n+1)(2n+7)/6$
f) $1/(1*2)+1/(2*3)+1/(3*4)\ldots+1/(n(n+1)) = n/(n+1)$
g) $1*1!+2*2!+3*3!\ldots+n*n! = (n+1)!-1$

Exercício 8.2 Prove por indução que:

a) Para qualquer natural $n > 1$, vale $n^2 > n+1$
b) Para qualquer natural $n > 6$, vale $n^2 > 5n+10$

Exercício 8.3 Prove por indução que, para qualquer $n \in \mathbf{N}$:

a) $2^{3n}-1$ é divisível por 7
b) $2^n+(-1)^{n+1}$ é divisível por 3

Exercício 8.4 Prove por indução que, qualquer número natural maior ou igual a dois pode ser definido como o produto de números primos.

Dica: use o segundo princípio de indução.

Exercício 8.5 Suponha que $A(n)$ denota $1+2+\ldots+n = ((2n+1)^2)/8$. Então:

a) Prove que, se $A(k)$ é verdadeiro para um $k \in \mathbf{N}$, então $A(k+1)$ também é verdadeiro;
b) Considerando o item acima, discuta a afirmação:

portanto, por indução, $A(n)$ é verdadeiro para qualquer $n \in \mathbf{N}$

c) De fato, $A(n)$ é verdadeiro para qualquer $n \in \mathbf{N}$? Prove a sua resposta.

Exercício 8.6 Por que a "prova por indução" que segue *não* é correta?

a) *Proposição*. Dado um conjunto de n torcedores de futebol, se pelo menos um torcedor é gremista, então todos os demais torcedores também são gremistas;
b) *"Prova"*. A proposição é trivialmente verdadeira para $n=1$. O passo de indução pode ser facilmente entendido pelo seguinte exemplo:
   - suponha que a proposição é verdadeira para $n=3$;
   - sejam T_1, T_2, T_3 e T_4, quatro torcedores dos quais pelo menos um é gremista (suponha que é T_1);
   - supondo o conjunto $\{T_1, T_2, T_3\}$ e a hipótese de que é verdadeiro para $n=3$, então T_2 e T_3 são gremistas;
   - analogamente para $\{T_1, T_2, T_4\}$, tem-se que T_2 e T_4 são gremistas;
   - portanto, os quatro torcedores são gremistas!
   - a generalização da construção acima para k e $k+1$, é a prova desejada.

Exercício 8.7 Para cada uma das definições indutivas abaixo, descreva o conjunto definido:

a) O conjunto A é indutivamente definido como segue:

$5 \in A$
se $x \in A$ e $y \in A$, então $x+y \in A$

b) O conjunto B é indutivamente definido como segue:

$2 \in B$ e $3 \in B$
se $b \in B$, então $2b \in B$ e $3b \in B$

Exercício 8.8 Considere o alfabeto constituído pelos símbolos das operações sobre conjuntos $\cup$ (união), $\cap$ (intersecção) e $\sim$ (complemento), pelos símbolos de parênteses, bem como por letras maiúsculas ($A, B, C, \ldots$) as quais denotam variáveis conjunto. Defina indutivamente as fórmulas bem formadas da álgebra de conjuntos usando esse alfabeto.

Exercício 8.9 Defina indutivamente o conjunto de todas as palíndromos sobre o alfabeto $\{a, b\}$.

Exercício 8.10 Relativamente à operação de exponenciação x^n, para um número x real:

a) Defina indutivamente a exponenciação em termos de multiplicações sucessivas;
b) Detalhe a aplicação da função definida para $n=3$;
b) Desenvolva uma correspondente função em Pascal (ou em outra linguagem de seu domínio).

Exercício 8.11 Relativamente à operação de multiplicação de dois números naturais:

a) Defina indutivamente a multiplicação de naturais em termos de adições sucessivas;
b) Detalhe a multiplicação de 2 e 3;
c) Desenvolva uma correspondente função em Pascal (ou em outra linguagem de seu domínio).

Exercício 8.12 Relativamente à função $\max: \mathbf{N}^2 \rightarrow \mathbf{N}$ a qual determina o maior de dois números fornecidos (por exemplo, $\max\langle 2, 3\rangle = 3$):

a) Defina indutivamente tal função;
b) Detalhe a aplicação da função definida para os valores 2 e 3;
c) Desenvolva uma correspondente função em Pascal (ou em outra linguagem de seu domínio).

Exercício 8.13 Considere a Definição 8.3 - Expressão Regular. Para cada expressão abaixo, verifique se é expressão regular e justifique a sua resposta:

a) $(a)^*$
b) $((a))$
c) $(a+a)^b$
d) $((a+\varepsilon)(ab))$
e) $((a+(bb))^*)$

Exercício 8.14 Considere a Definição 8.3 - Expressão Regular. Supondo que o item referente à concatenação no passo de indução é reescrito como segue (observe que a concatenação passa a ser denotada usando o símbolo • e não mais pela justaposição):

$(r \bullet s)$ é expressão regular e denota a linguagem $RS = \{u \bullet v \mid u \in R \text{ e } v \in S\}$

demonstre que toda expressão regular assim definida (explicitamente representando a concatenação usando o símbolo •) possui comprimento (número de símbolos) ímpar.

Sugestão: use o Segundo Princípio da Indução Matemática.

Exercício 8.15 Considere o alfabeto $\Sigma = \{a, b\}$.

a) Desenvolva expressões regulares que definam as seguintes linguagens:

a.1) $\{w \mid w \text{ possui } aaa \text{ como subpalavra}\}$
a.2) $\{w \mid \text{o quinto símbolo da direita para a esquerda de } w \text{ é } a\}$

b) Descreva em palavras as linguagens definidas pelas seguintes expressões regulares:

b.1) $(aa+b)^*(a+bb)$

b.2) $(b+ab)^*(\varepsilon+a)$

Exercício 8.16 Relativamente ao EXEMPLO 8.14 - Função Programa Estendida, calcule a computação do autômato finito para a palavra de entrada baba:

a) A partir do estado q_0;

b) A partir do estado q_2.

Exercício 8.17 Relativamente ao EXEMPLO 8.15 - Gramática e Derivação: Números Naturais:

a) Existe mais alguma derivação do número 243? Nesse caso, quantas?

b) Modifique a gramática de tal forma a não distinguir zeros à esquerda.

Exercício 8.18 Relativamente ao EXEMPLO 8.16 - Gramática e Derivação: Expressão Aritmética:

a) Identifique, claramente, em cada passo de derivação, qual a regra de produção usada para gerar a palavra $(x+x)*x$;

b) Existe mais alguma derivação da palavra $(x+x)*x$? Nesse caso, quantas?

c) Para o conjunto de regras apresentado, qual a correspondente definição indutiva?

Exercício 8.19 Para cada uma das linguagens abaixo, desenvolva uma gramática que gere a linguagem:

a) $L_1 = \varnothing$

b) $L_2 = \{\varepsilon\}$

c) $L_3 = \{a, b\}^*$

d) $L_4 = \{w \mid w \text{ é palíndromo em } \{a, b\}^*\}$

Exercício 8.20 Considere o EXEMPLO 8.17 - BNF: Identificador em Pascal. Por que a seguinte BNF não é equivalente à apresentada no exemplo?

⟨identificador⟩ ::= ⟨letra⟩ | ⟨letra⟩⟨identificador⟩ | ⟨dígito⟩⟨identificador⟩
⟨letra⟩ ::= a | b | ... | z
⟨dígito⟩ ::= 0 | 1 | ... | 9 ❑

Exercício 8.21 Usando a linguagem de programação Pascal (ou outra de seu conhecimento), para cada um dos itens abaixo, desenvolva uma correspondente BNF:

a) Identificadores com, no máximo, 6 caracteres;

b) Números inteiros (com ou sem sinal);

c) Números com ponto flutuante (em Pascal, números reais).

Exercício 8.22 Uma florista é famosa pelos belos ramos de rosas que arruma. Os ramos são compostos por rosas e galhos de folhas (denominados simplesmente de folhas). Na confecção dos ramos a florista obedece às 4 regras descritas abaixo. A partir dessas regras defina indutivamente um ramo de rosas:

- o número de rosas deve ser sempre ímpar;
- o número de folhas deve ser sempre par;
- todo ramo tem que ter folhas;
- o número de folhas pode ser menor que o número de rosas, e não pode ultrapassar por mais de um o número de rosas.

Exercício 8.23 Defina indutivamente o seguinte problema:

Um jarro tem p bolas pretas e b bolas brancas. Tiram-se duas bolas quaisquer e:

- se forem de mesma cor, coloca-se uma bola preta na jarra (existe um estoque de bolas pretas, caso seja necessário);
- se forem de cores diferentes, coloca-se uma bola branca na jarra.

Exercício 8.24 Existe uma versão finita do Princípio da Indução Matemática a qual, analogamente à infinita, pode ser aplicada a provas e definições. A versão finita é definida como segue:

Seja $p(m), p(m+1), \ldots, p(n)$, uma seqüência de proposições. O *Princípio da Indução Matemática Finita* é como segue:

a) $p(m)$ é verdadeira;

b) Para qualquer $k \in \mathbf{N}$ tal que $m \le k < n$ vale $p(k) \Rightarrow p(k+1)$

c) Então, para qualquer $k \in \mathbf{N}$ tal que $m \le k < n$, $p(k)$ é verdadeira.

Usando o Princípio da Indução Matemática Finita, defina indutivamente uma Microempresa, sabendo que:

a) As regras que definem uma microempresa são as seguintes:

- tem que ter pelo menos 10 e no máximo 100 empregados;
- tem que produzir pelo menos um produto;

b) As operações sobre a constituição da microempresa são as seguintes:

- contratar um empregado;
- demitir um empregado;
- produzir um novo produto;
- deixar de produzir um produto.

Exercício 8.25 Usando as funções recursivas parciais $const_{zero}$, suc e ad, verifique se existem outras formas de definir equivalentemente as funções $const_{um}$, $const_{dois}$ e $const_{três}$ apresentadas no EXEMPLO 8.22 - Substituição de Funções: Funções Constantes.

Exercício 8.26 Compare e diferencie claramente as funções $const_{zero}$ (EXEMPLO 8.22 - Substituição de Funções: Funções Constantes) e zero (EXEMPLO 8.24 - Minimização, Recursão: Número Zero).

Exercício 8.27 Considere o EXEMPLO 8.26 - Minimização, Recursão: Subtração. Determine o valor de $sub\langle 2, 3\rangle$.

Exercício 8.28 Desenvolva funções recursivas parciais sobre **N** para as seguintes operações:

a) Multiplicação;

b) Quadrado (n^2);

c) Fatorial ($n!$).

Sugestão: use a função recursiva de adição definida nos exemplos.

Exercício 8.29 *Função de Ackernmann.* A função de Ackernmann é um importante exemplo no estudo das funções recursivas e das funções *Recursivas Primitivas* (funções recursivas, mas totais e sem minimização). Historicamente, foi considerada a possibilidade de que as funções recursivas primitivas definissem a Classe das Funções Totais Computáveis. Entretanto, a função de Ackernmann (prova-se) é um exemplo de função recursiva (total) a qual não é primitiva.

A função de Ackernmann $\mathsf{ack}: \mathbf{N}^2 \rightarrow \mathbf{N}$ é tal que:

$$\mathsf{ack}\langle 0, y\rangle = 1$$
$$\mathsf{ack}\langle 1, 0\rangle = 2$$
$$\mathsf{ack}\langle x, 0\rangle = x + 2, \text{ para } x \geq 2$$
$$\mathsf{ack}\langle x+1, y+1\rangle = \mathsf{ack}\langle \mathsf{ack}\langle x, y+1\rangle, y\rangle$$

a) A definição acima satisfaz a definição de função recursiva?

b) Calcule, passo a passo:

$$\mathsf{ack}\langle 0, 0\rangle = 1$$
$$\mathsf{ack}\langle 2, 1\rangle = 1$$

c) Defina, formalmente, a função recursiva de um argumento $\mathsf{ack}\langle x, 2\rangle$.

9 Álgebras e Homomorfismos

Relativamente às *álgebras*, já foi comentado que:

a) *Álgebra*, desde a sua origem até a sua forma atual, refere-se a cálculos. Com limitações, é desenvolvida de maneira informal ou formal, em praticamente todos os níveis de escolaridade. Por exemplo, as operações aritméticas básicas (adição, multiplicação, etc.) sobre o conjunto dos números reais constituem uma álgebra;

b) Historicamente, o estudo das álgebras, em Computação e Informática, destaca-se a partir de 1950, com o desenvolvimento da Teoria dos Autômatos e Linguagens Formais. De certa forma, toda a Computação e Informática é baseada, direta ou indiretamente, sobre álgebras;

c) As Diretrizes Curriculares do MEC para Cursos de Computação e Informática [MEC 2004] referem-se à Álgebra como sendo uma denominação alternativa para a Matemática Discreta.

Assim, o estudo mais amplo de álgebras é central no contexto da Matemática Discreta. De fato, alguns exemplos de álgebras já foram apresentados, explicita ou implicitamente:

- *Álgebra de Conjuntos* constituída por todos os conjuntos e as operações sobre conjuntos como união, produto cartesiano, etc.;
- *Álgebra de Proposições* constituída por todas as proposições e conetivos lógicos (e, negação, etc.);
- *Álgebra de Funções* constituída por todas as funções e a operação de composição de funções. Seguindo a mesma linha de raciocínio, existe, ainda, a *Álgebra de Relações*, a *Álgebra de Funções Parciais*, etc.

O conceito formal de álgebra é relativamente simples. Entretanto, a caracterização de todos os tipos de álgebras possíveis, resulta em um conceito geral e, portanto, com um nível de abstração relativamente alto. Assim, apresentar o conceito formal de álgebra e depois exemplificar com alguns tipos de especial interesse para Computação e Informática pode ser pouco produtivo. Portanto, optou-se por apresentar inicialmente alguns exemplos de estruturas algébricas, construindo-se o conceito de álgebra do concreto para o abstrato.

Casualmente, todos os exemplos de álgebras acima são de *álgebras grandes*, pois são constituídas de operações definidas sobre coleções de objetos (todos os conjuntos, todas as funções, etc.) os quais não são conjuntos. Neste capítulo, é dada ênfase às *álgebras pequenas*, ou seja, álgebras definidas sobre conjuntos.

Um conceito tão importante quanto o de álgebra é o de *homomorfismo de álgebras*, o qual é constituído por funções (no caso de álgebras pequenas) que mapeiam álgebras (estruturalmente similares), preservando as suas estruturas. De fato, o termo *morfismo* é usado genericamente para representar alguma forma de mapeamento (relação, função, etc.) entre duas estruturas similares, o qual, quando prefixado por *homo*, destaca que o morfismo em questão preserva a estrutura. A noção de homomorfismo é desenvolvida gradativamente ao longo deste capítulo, para cada estrutura algébrica exemplificada.

Para o desenvolvimento de um estudo mais formal de álgebra, é necessário primeiramente apresentar o conceito de operação e as correspondentes principais propriedades.

Neste capítulo, são discutidas algumas importantes construções baseadas em álgebras e os correspondentes homomorfismos tais como *fecho de Kleene*, *grafo* (visto como álgebra) e *categoria* (também visto como álgebra).

9.1 Operações Binárias

Já foi comentado que o termo *operação* (pequena) é sinônimo de função parcial. De especial interesse para a Computação e Informática são os seguintes tipos de operações:

- *operações binárias*, ou seja, cujo domínio é um conjunto resultante de um produto cartesiano;
- *operações internas* a um conjunto A, ou seja, cujo domínio e contradomínio são definidos sobre A;
- *operações fechadas*, ou seja, total.

Definição 9.1 - Operação Binária, Operação Interna, Operação Fechada
Sejam A, B e C conjuntos. Então:

a) Uma *Operação Binária* é uma função parcial do tipo:

$$\oplus: A \times B \rightarrow C$$

b) Uma *Operação Interna* ao conjunto A é uma operação cujo domínio e contradomínio são definidos em A (o próprio conjunto A ou o conjunto resultante do produto cartesiano sobre A). Em particular, uma *operação binária interna* ao conjunto A é uma operação do tipo:

$$\oplus: A^2 \rightarrow A$$

c) Uma *Operação Fechada* é uma operação total (ou seja, é uma função). ❑

EXEMPLO 9.1 - Operação

a) *Divisão nos reais*. A operação $\text{div}: \mathbf{R}^2 \rightarrow \mathbf{R}$, definida como abaixo (suponha $\langle x, y\rangle \in \mathbf{R} \times \mathbf{R}$), é uma operação binária interna a **R** (é fechada?):

$$\text{div}\langle x, y\rangle = x/y$$

b) *Quadrado nos naturais*. A operação $\text{quadrado}: \mathbf{N} \rightarrow \mathbf{N}$, definida como abaixo (suponha $n \in \mathbf{N}$), é uma operação interna e fechada:

$$\text{quadrado}\langle n\rangle = n^2$$

c) *Elemento zero*. Seja $1 = \{*\}$ um conjunto unitário. A operação $\text{zero}: 1 \rightarrow \mathbf{N}$, definida como abaixo, a qual identifica o número zero nos naturais, é uma operação fechada:

$$\text{zero}\langle * \rangle = 0$$

d) *União*. Seja A um conjunto qualquer. A operação $\cup: \mathbf{P}(A) \times \mathbf{P}(A) \rightarrow \mathbf{P}(A)$ é uma operação binária interna e fechada. ❑

9.2 Propriedades das Operações Binárias

As principais propriedades das operações binárias, internas e fechadas são as seguintes:

- *comutativa*;
- *associativa*;
- *elemento neutro*;
- *elemento inverso*.

As propriedades comutativa, associativa e elemento neutro já foram apresentadas anteriormente quando do estudo da Álgebra de Conjuntos (algumas noções também foram apresentadas quando do estudo da Lógica).

Definição 9.2 - Comutativa, Associativa, Elemento Neutro, Elemento Inverso
Seja $\oplus: A^2 \rightarrow A$ uma operação binária, interna e fechada. Então a operação $\oplus$ satisfaz as propriedades:

a) *Comutativa*:

$$(\forall a \in A)(\forall b \in A)\,(a \oplus b = b \oplus a)$$

b) *Associativa*:

$$(\forall a \in A)(\forall b \in A)(\forall c \in A)\,(a \oplus (b \oplus c) = (a \oplus b) \oplus c)$$

c) *Elemento Neutro*:

$$(\exists e \in A)(\forall a \in A)\,(a \oplus e = e \oplus a = a)$$

d) *Elemento Inverso*:

$$(\forall a \in A)(\exists \underline{a} \in A)\,(a \oplus \underline{a} = \underline{a} \oplus a = e)$$ ❑

Portanto, uma operação satisfaz as propriedades:

a) *Comutativa*, se a ordem de operação dos operandos não é importante. Neste caso, afirma-se que a operação é comutativa;

b) *Associativa*, se a precedência na aplicação do operador não é importante e, portanto, os parênteses podem ser omitidos, ou seja, $a \oplus (b \oplus c)$ ou $(a \oplus b) \oplus c$ pode, equivalentemente, ser denotado sem parênteses como segue:

$$a \oplus b \oplus c$$

Neste caso, afirma-se que a operação é associativa;

c) *Elemento Neutro*, se existe um elemento neutro o qual, operado à *direita* $(a \oplus e)$ bem como à *esquerda* $(e \oplus a)$, resulta sempre no outro operando. Neste caso, afirma-se que a operação possui elemento neutro;

d) *Elemento Inverso*, se cada elemento tem elemento inverso. Neste caso, afirma-se que a operação tem inverso.

Observação 9.3 - Elemento Neutro à Esquerda e à Direita
Observe que o fato de satisfazer à propriedade elemento neutro *simultaneamente* à esquerda e à direita é fundamental. De fato, existem operações que possuem elemento neutro em somente um dos casos (e, portanto, não possuem elemento neutro), como, por exemplo, a divisão nos reais: possui neutro à direita (o número um - qualquer número dividido por um resulta no próprio) mas não possui neutro à esquerda. ❑

EXEMPLO 9.2 - Propriedades da União
Seja A um conjunto. Já foi visto que a operação de união $\cup: \mathbf{P}(A) \times \mathbf{P}(A) \rightarrow \mathbf{P}(A)$ satisfaz as propriedades comutativa, associativa e de elemento neutro (o qual é o conjunto vazio). ❑

EXEMPLO 9.3 - Propriedades da Adição

a) A operação de adição nos naturais $+: \mathbf{N}^2 \rightarrow \mathbf{N}$ satisfaz as propriedades comutativa, associativa e de elemento neutro (o qual é o zero);

b) Assim como a adição nos naturais, a adição nos inteiros $+: \mathbf{Z}^2 \rightarrow \mathbf{Z}$ satisfaz as propriedades comutativa, associativa e de elemento neutro. Adicionalmente, satisfaz a propriedade de elemento inverso. De fato, para qualquer inteiro n, basta tomar ^-n como elemento inverso:

$$n + {}^-n = {}^-n + n = 0$$ ❑

EXEMPLO 9.4 - Propriedades da Multiplicação

a) A operação de multiplicação nos naturais $*: \mathbf{N}^2 \rightarrow \mathbf{N}$ satisfaz as propriedades comutativa, associativa e de elemento neutro (o qual é o um);

b) Da mesma forma, a operação de multiplicação nos reais $*: \mathbf{R}^2 \rightarrow \mathbf{R}$ também satisfaz as propriedades comutativa, associativa e de elemento neutro. Entretanto, se for considerada a multiplicação nos reais *sem o zero*, então a operação também satisfaz a propriedade de elemento inverso. De fato, para qualquer real x, basta tomar 1/x como elemento inverso, ou seja (por que sem o zero?):

$$x * 1/x = 1/x * x = 1$$ ❑

9.3 Grupóides, Semigrupos, Monóides, Grupos

Uma operação binária e interna $\oplus: A^2 \rightarrow A$ é um exemplo de *álgebra* e é usualmente denotada como um par ordenado como segue:

$$\langle A, \oplus \rangle$$

Em particular, como a operação $\oplus$ é interna, a álgebra é denominada de *álgebra interna*. Em uma álgebra interna $\langle A, \oplus \rangle$, o conjunto A é usualmente denominado de *conjunto suporte*.

Como já destacado, as operações binárias e internas são especialmente importantes para Computação e Informática. Os tipos mais importantes de álgebras internas com uma única operação binária (interna), juntamente com as suas correspondentes propriedades, são brevemente apresentados na Figura 9.1. Adicionalmente, se a operação da álgebra for comutativa, então esta é dita uma álgebra comutativa ou *abeliana*. Por exemplo, um grupóide $\langle A, \oplus \rangle$ tal que a operação $\oplus$ é comutativa é denominado de *grupóide abeliano*.

Tipo de Álgebra Interna	Fechada	Associativa	Elemento Neutro	Elemento Inverso
Grupóide	✔			
Semigrupo	✔	✔		
Monóide	✔	✔	✔	
Grupo	✔	✔	✔	✔

Figura 9.1 Tipos de álgebras e as correspondentes propriedades

Conseqüentemente, existe uma hierarquia entre esses tipos de álgebras, como ilustrado na Figura 9.2, a qual representa inclusões próprias. Assim, por exemplo:

- todo grupóide é uma álgebra interna, mas nem toda álgebra interna é um grupóide;
- todo grupo é um monóide (respectivamente, semigrupo, grupóide), mas nem todo monóide (respectivamente, semigrupo, grupóide) é um grupo.

Definição 9.4 - Grupóide, Semigrupo, Monóide, Grupo

Seja $\oplus: A^2 \rightarrow A$ uma operação (binária e interna). Então $\langle A, \oplus \rangle$ é um:

a) *Grupóide*, se a operação for fechada;

b) *Semigrupo*, se $\langle A, \oplus \rangle$ é um grupóide e, adicionalmente, a operação $\oplus: A^2 \rightarrow A$ é associativa;

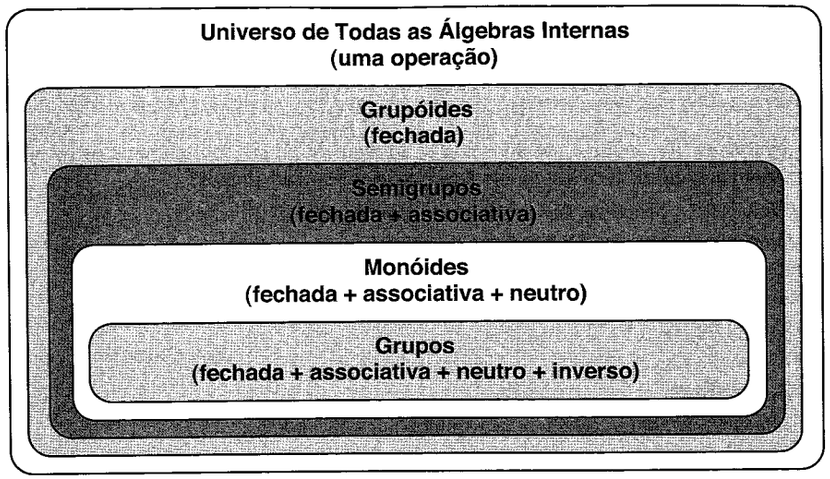


Figura 9.2 Hierarquia das álgebras internas

c) *Monóide*, se $\langle A, \oplus \rangle$ é um semigrupo e, adicionalmente, a operação $\oplus: A^2 \rightarrow A$ satisfaz a propriedade de elemento neutro. Neste caso, com o objetivo de destacar o elemento neutro, o monóide é freqüentemente denotado como segue (supondo que $e \in A$ é o elemento neutro):

$$\langle A, \oplus, e \rangle$$

d) *Grupo*, se $\langle A, \oplus, e \rangle$ é um monóide e, adicionalmente, a operação $\oplus: A^2 \rightarrow A$ satisfaz a propriedade de elemento inverso.

Adicionalmente, se a operação for comutativa, então o grupóide, semigrupo, monóide ou grupo é dito, respectivamente:

- *Grupóide Comutativo* ou *Grupóide Abeliano*;
- *Semigrupo Comutativo* ou *Semigrupo Abeliano*;
- *Monóide Comutativo* ou *Monóide Abeliano*;
- *Grupo Comutativo* ou *Grupo Abeliano*. ❑

EXEMPLO 9.5 - Grupóide, Semigrupo, Monóide: Concatenação

Seja Σ um alfabeto não-vazio. Considere a seguinte operação de concatenação:

$$\text{conc}: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

a qual é fechada, associativa e possui elemento neutro (o qual é a palavra vazia ε). Portanto, a álgebra interna $\langle \Sigma^*, \text{conc} \rangle$ é simultaneamente um grupóide, um semigrupo e um monóide. Adicionalmente, em geral (procure justificar):

- não é um grupo (e se o alfabeto for vazio?);
- não é uma álgebra comutativa (e se o alfabeto for vazio ou unitário?). ❑

EXEMPLO 9.6 - Grupóide, Semigrupo, Monóide: União e Intersecção

Seja A um conjunto. Considere as seguintes operações de união e de intersecção, respectivamente:

$$\cup: \mathbf{P}(A) \times \mathbf{P}(A) \rightarrow \mathbf{P}(A) \quad \text{e} \quad \cap: \mathbf{P}(A) \times \mathbf{P}(A) \rightarrow \mathbf{P}(A)$$

as quais (já foi visto) são fechadas, associativas, possuem elemento neutro (os quais são, respectivamente, os conjuntos $\varnothing$ e A) e comutativas. Então, as seguintes álgebras internas são simultaneamente grupóides abelianos, semigrupos abelianos e monóides abelianos:

$$\langle \mathbf{P}(A), \cup \rangle \quad \text{e} \quad \langle \mathbf{P}(A), \cap \rangle$$

Se as operações de união e de intersecção fossem definidas sobre todos os conjuntos, constituiriam grupóides? E se o conjunto A for vazio, constituem grupos? ❑

EXEMPLO 9.7 - Grupóide, Semigrupo, Monóide, Grupo: Adição e Multiplicação

Considerando as operações de adição e de multiplicação, as seguintes álgebras internas são simultaneamente grupóides abelianos, semigrupos abelianos e monóides abelianos (qual o elemento neutro em cada caso?):

$\langle \mathbf{N}, + \rangle$ e $\langle \mathbf{N}, * \rangle$
$\langle \mathbf{Z}, + \rangle$ e $\langle \mathbf{Z}, * \rangle$
$\langle \mathbf{R}, + \rangle$ e $\langle \mathbf{R}, * \rangle$

Adicionalmente, são grupos abelianos as seguintes álgebras internas:

$\langle \mathbf{Z}, + \rangle$
$\langle \mathbf{R}, + \rangle$
$\langle \mathbf{R} - \{0\}, * \rangle$ ❑

EXEMPLO 9.8 - Grupóide, Semigrupo, Monóide, Grupo: Unitário

A seguinte álgebra interna é simultaneamente grupóide abeliano, semigrupo abeliano, monóide abeliano e grupo abeliano:

$$\langle \{*\}, ! \rangle$$

supondo que a operação $!: \{*\} \times \{*\} \rightarrow \{*\}$ é fechada, então é associativa, comutativa, possui elemento neutro (o qual é o único elemento do suporte), possui elemento inverso (por quê?) e é a única com origem em $\{*\} \times \{*\}$ e destino em $\{*\}$ (por quê?).

Observe que o menor monóide (em termos do cardinal do conjunto suporte) é o unitário. ❑

EXEMPLO 9.9 - Grupóide, Semigrupo: Vazio

A seguinte álgebra interna é simultaneamente grupóide abeliano e semigrupo abeliano:

$$\langle \varnothing, \varnothing \rangle$$

sendo que a operação vazia $\varnothing: \varnothing \times \varnothing \rightarrow \varnothing$ é fechada, associativa (por quê?), comutativa e é a única com origem em $\varnothing \times \varnothing$ e destino em $\varnothing$ (por quê?).

Observe que o menor grupóide (em termos do cardinal do conjunto suporte) é o vazio. ❑

EXEMPLO 9.10 - Álgebra Não-Grupóide

As seguintes álgebras internas *não* são grupóides (procure justificar cada um dos itens):

Subtração nos naturais: $\langle \mathbf{N}, - \rangle$
Divisão nos reais: $\langle \mathbf{R}, / \rangle$
Produto cartesiano no conjunto das partes: $\langle \mathbf{P}(A), \times \rangle$, supondo A não-vazio ❑

EXEMPLO 9.11 - Álgebra Não-Semigrupo

As seguintes álgebras internas *são* grupóides, mas *não* são semigrupos, pois as respectivas operações não são associativas (procure justificar cada um dos itens):

Subtração nos inteiros: $\langle \mathbf{Z}, - \rangle$
Divisão nos reais sem o zero: $\langle \mathbf{R} - \{0\}, / \rangle$ ❑

EXEMPLO 9.12 - Álgebra Não-Monóide

a) *Vazio*. O semigrupo abeliano $\langle \varnothing, \varnothing \rangle$ (EXEMPLO 9.9 - Grupóide, Semigrupo: Vazio), claramente não possui elemento neutro. Lembre-se de que o elemento neutro é um elemento do conjunto suporte, e, portanto, o conjunto suporte de um monóide *não* pode ser vazio;

b) *Adição e Multiplicação*. As operações de adição e de multiplicação sobre conjuntos, excluindo-se, respectivamente, os elementos 0 e 1, obviamente não constituem monóides, como, por exemplo:

$\langle \mathbf{N} - \{0\}, + \rangle$ e $\langle \mathbf{N} - \{1\}, * \rangle$
$\langle \mathbf{R} - \{0\}, + \rangle$ e $\langle \mathbf{R} - \{1\}, * \rangle$ ❑

EXEMPLO 9.13 - Álgebra Não-Grupo

Procure justificar por que cada uma das seguintes álgebras *não* é um grupo:

a) *Adição e Multiplicação*.

$\langle \mathbf{N}, + \rangle$ adição nos inteiros
$\langle \mathbf{R}, * \rangle$ multiplicação nos reais

b) *União e Intersecção*. Seja A um conjunto *não-vazio*.

$\langle \mathbf{P}(A), \cup \rangle$
$\langle \mathbf{P}(A), \cap \rangle$

c) *Concatenação*. Seja Σ um alfabeto *não-vazio*.

$\langle \Sigma^*, \text{conc}, \varepsilon \rangle$ ❑

9.4 Importantes Propriedades dos Monóides e Grupos

O seguinte teorema mostra que o elemento neutro em um monóide (e, conseqüentemente, em um grupo) é único. De fato, o teorema, com a correspondente prova, é uma generalização do seguinte resultado apresentado quando do estudo das Técnicas de Demonstração (prova por absurdo):

0 é o único elemento neutro da adição em $\mathbf{N}$

Teorema 9.5 - Elemento Neutro de um Monóide é Único

Seja $\langle A, \oplus, e \rangle$ um monóide. Então $e \in A$ é o único elemento neutro do monóide.

Prova: (*por absurdo*)

Seja $\langle A, \oplus, e \rangle$ um monóide. Suponha que e *não* é o único elemento neutro. Portanto, existe um outro elemento neutro $\underline{e} \neq e$. Então:

- como e é elemento neutro, para qualquer $a \in A$, tem-se que $a = e + a = a + e$. Em particular, para $a = \underline{e}$, tem-se que:

$$\underline{e} = e + \underline{e} = \underline{e} + e \qquad (1)$$

- como $\underline{e}$ é elemento neutro, para qualquer $a \in A$, tem-se que $a = \underline{e} + a = a + \underline{e}$. Em particular, para $a = e$, tem-se que:

$$e = \underline{e} + e = e + \underline{e} \qquad (2)$$

- portanto, pela transitividade da igualdade em (1) e (2), tem-se que $\underline{e} = e$, o que é uma contradição, pois foi suposto que $\underline{e} \neq e$

Assim, é absurdo supor que o elemento neutro do monóide $\langle A, \oplus, e\rangle$ não é único. Logo, o elemento neutro de qualquer monóide é único. ❑

O seguinte Teorema apresenta uma importante propriedade para os grupos, a qual generaliza a intuição sobre operações como a adição nos reais (bem como nos inteiros ou racionais). Por exemplo, sabe-se que, para a seguinte equação nos reais:

$$x+3=y+3$$

obtem-se $x=y$. Tal propriedade é usualmente denominada de *cancelamento*.

Teorema 9.6 - Propriedade de Cancelamento dos Grupos
Seja $\langle A, \oplus, e\rangle$ um grupo. Então, a *propriedade de cancelamento* é satisfeita, ou seja, simultaneamente:

a) *Cancelamento à direita*:

$$(\forall a \in A)(\forall x \in A)(\forall y \in A)\,(x \oplus a = y \oplus a \rightarrow x = y)$$

b) *Cancelamento à esquerda*:

$$(\forall a \in A)(\forall x \in A)(\forall y \in A)\,(a \oplus x = a \oplus y \rightarrow x = y)$$ ❑

Prova:
Suponha $\langle A, \oplus, e\rangle$ um grupo. Para quaisquer $a \in A$, $x \in A$ e $y \in A$, vale:

a) *Cancelamento à direita*. Suponha que $x \oplus a = y \oplus a$. Então:

$x =$	elemento neutro
$x \oplus e =$	elemento inverso
$x \oplus (a \oplus \underline{a}) =$	associatividade
$(x \oplus a) \oplus \underline{a} =$	hipótese
$(y \oplus a) \oplus \underline{a} =$	associatividade
$y \oplus (a \oplus \underline{a}) =$	elemento inverso
$y \oplus e =$	elemento neutro
y	

Portanto, $x = y$

b) *Cancelamento à esquerda*. A prova é análoga e é sugerida como exercício.

Logo, a propriedade de cancelamento é satisfeita. ❑

A prova do seguinte teorema é sugerida como exercício.

Teorema 9.7 - Elemento Inverso em um Grupo é Único
Seja $\langle A, \oplus\rangle$ um grupo. Então, para qualquer $a \in A$, o elemento inverso de a é único. ❑

9.5 Homomorfismos

Já foi afirmado que um homomorfismo de álgebras é constituído por funções (no caso de álgebras pequenas) que mapeiam álgebras de um mesmo tipo, preservando as suas estruturas. No que segue, é estudado como as estruturas de grupóides, semigrupos, monóides e grupos (abelianos ou não) são preservadas por homomorfismos.

Como homomorfismo de álgebras é constituído por funções, seria de se esperar que os conceitos de monomorfismo, de epimorfismo e de isomorfismo fossem naturalmente estendidos

para mapeamentos de álgebras. Entretanto, somente o de isomorfismo pode ser estendido, como destacado na importante observação a seguir.

Observação 9.8 - Monomorfismo, Epimorfismo e Isomorfismo de Álgebras
O conceito de *isomorfismo*, baseado na existência de um morfismo inverso, pode ser naturalmente estendido para as estruturas algébricas em geral, conforme é destacado no texto que segue. Entretanto, a correta extensão dos conceitos de *monomorfismo* e de *epimorfismo* exige noções e conceitos baseados em *Teoria das Categorias*, os quais fogem do escopo deste livro. Assim, o leitor deve ficar atento ao fato de que nem todo monomorfismo (respectivamente epimorfismo) entre estruturas algébricas é induzido por um monomorfismo (respectivamente, epimorfismo) entre conjuntos. ❑

Duas estruturas algébricas são ditas *isomorfas* se existe um isomorfismo entre tais estruturas. Nesse caso, as estruturas são consideradas basicamente a mesma, ou seja, *iguais a menos de isomorfismo* e, obrigatoriamente, possuem as mesmas propriedades.

9.5.1 Homomorfismo de Grupóides e de Semigrupos

Um *homomorfismo de grupóides* é constituído por uma função entre os conjuntos suportes tal que preserva a operação no seguinte sentido (veja a Figura 9.3):

a imagem do resultado da operação dos operandos no grupóide origem
é igual ao resultado da operação da imagem dos operandos no grupóide destino.

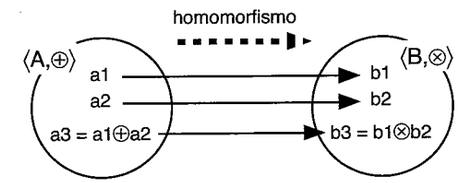


Figura 9.3 Homomorfismo de grupóides preserva a operação

Um *homomorfismo de semigrupos* é como um homomorfismo de grupóides, no sentido em que basta preservar a operação.

Definição 9.9 - Homomorfismo de Grupóides
Sejam $\langle A, \oplus\rangle$ e $\langle B, \otimes\rangle$ dois grupóides. Um *Homomorfismo de Grupóides*:

$$h: \langle A, \oplus\rangle \rightarrow \langle B, \otimes\rangle$$

é uma função entre os conjuntos suportes $h: A \rightarrow B$ tal que:

$$(\forall a_1 \in A)(\forall a_2 \in A)(\, h\langle a_1 \oplus a_2\rangle = h\langle a_1\rangle \otimes h\langle a_2\rangle\,)$$ ❑

A notação $h: \langle A, \oplus\rangle \rightarrow \langle B, \otimes\rangle$ (e não simplesmente como uma função $h: A \rightarrow B$) destaca o fato de que se trata de um morfismo entre álgebras.

Alguns homomorfismos de grupóides são óbvios como, por exemplo:

- identidade;
- inclusão.

Entretanto, nem sempre um homomorfismo é óbvio.

EXEMPLO 9.14 - Homomorfismo de Grupóides: Identidade, Inclusão
Sejam $\langle A, \oplus\rangle$, $\langle \mathbf{N}, +\rangle$ e $\langle \mathbf{Z}, +\rangle$ grupóides. Então:

a) *Identidade.* A função identidade $id_A: A \to A$ induz o homomorfismo identidade de grupóides:

$$id_{\langle A,\oplus\rangle}: \langle A, \oplus\rangle \to \langle A, \oplus\rangle$$

b) *Inclusão.* A função inclusão $inc_{\mathbf{N},\mathbf{Z}}: \mathbf{N} \to \mathbf{Z}$ induz o homomorfismo inclusão de grupóides:

$$inc_{\langle \mathbf{N},+\rangle,\langle \mathbf{Z},+\rangle}: \langle \mathbf{N}, +\rangle \to \langle \mathbf{Z}, +\rangle$$ ❑

Observe que, no EXEMPLO 9.14 - Homomorfismo de Grupóides: Identidade, Inclusão, os morfismos apresentados consideram sempre a mesma operação. Ou seja, os termos identidade e inclusão consideram *toda a estrutura* do grupóide, e não apenas o conjunto suporte. O seguinte exemplo ilustra um caso em que, ao ser considerado apenas o conjunto suporte, a função identidade induz um morfismo o qual não é um homomorfismo de grupóides.

EXEMPLO 9.15 - Morfismo Não-Homomorfismo de Grupóides
Seja A um conjunto. Considere os grupóides $\langle \mathbf{P}(A), \cup\rangle$, $\langle \mathbf{P}(A), \cap\rangle$ e a função identidade $id_{\mathbf{P}(A)}: \mathbf{P}(A) \to \mathbf{P}(A)$. O seguinte morfismo induzido pela função $id_{\mathbf{P}(A)}$ *não* é, em geral, um homomorfismo de grupóides:

$$id_{\mathbf{P}(A)}: \langle \mathbf{P}(A), \cup\rangle \to \langle \mathbf{P}(A), \cap\rangle$$

Considere, por exemplo, $A = \{a, b\}$, sendo que $\mathbf{P}(A) = \{\varnothing, \{a\}, \{b\}, \{a, b\}\}$. Em particular:

$$id_{\mathbf{P}(A)}\langle\{a\} \cup \{b\}\rangle = id_{\mathbf{P}(A)}\langle\{a, b\}\rangle = \{a, b\}$$

o que é diferente de:

$$id_{\mathbf{P}(A)}\langle\{a\} \cup \{b\}\rangle = id_{\mathbf{P}(A)}\langle\{a\}\rangle \cap id_{\mathbf{P}(A)}\langle\{b\}\rangle = \{a\} \cap \{b\} = \varnothing$$

e, portanto, $id_{\mathbf{P}(A)}: \langle \mathbf{P}(A), \cup\rangle \to \langle \mathbf{P}(A), \cap\rangle$ não preserva a operação. Logo, não é homomorfismo de grupóides. ❑

De forma análoga, sugere-se como exercício apresentar um caso em que uma função inclusão (não-identidade) induz um morfismo o qual não é homomorfismo de grupóides.

O seguinte exemplo é especialmente importante no contexto da Computação e Informática.

EXEMPLO 9.16 - Homomorfismo de Grupóides: Concatenação
Sejam $\Sigma_1 = \{a, b, c\}$ e $\Sigma_2 = \{r, s\}$ alfabetos. Considere os grupóides $\langle \Sigma_1^*, conc_1\rangle$ e $\langle \Sigma_2^*, conc_2\rangle$. Então:

a) A função $f: \Sigma_1 \to \Sigma_2$ tal que (observe que é uma função entre *alfabetos*):

$f\langle a\rangle = r$
$f\langle b\rangle = r$
$f\langle c\rangle = s$

induz (canonicamente) o seguinte homomorfismo de grupóides (observe que mapeia *palavras*), o qual é ilustrado na Figura 9.4 (esquerda):

$$f^*: \langle \Sigma_1^*, conc_1\rangle \to \langle \Sigma_2^*, conc_2\rangle$$

o qual é indutivamente definido como segue:

$f^*\langle\varepsilon\rangle = \varepsilon$
para qualquer símbolo $x \in \Sigma_1$, vale $f^*\langle x\rangle = f\langle x\rangle$
se $x \in \Sigma_1$ e $w \in \Sigma_1^*$, então $f^*\langle x\,w\rangle = f\langle x\rangle\, f^*\langle w\rangle$

Por exemplo:

$f^*\langle abc\rangle =$

$f\langle a\rangle\, f^*\langle bc\rangle =$
$f\langle a\rangle\, f\langle b\rangle\, f^*\langle c\rangle =$
$f\langle a\rangle\, f\langle b\rangle\, f\langle c\rangle\, f^*\langle\varepsilon\rangle =$
$r\,r\,s\,\varepsilon = r\,r\,s$

b) A função entre alfabetos $g: \Sigma_2 \to \Sigma_1$ tal que:

$g\langle r\rangle = a$
$g\langle s\rangle = b$

induz o seguinte homomorfismo de grupóides, o qual é ilustrado na Figura 9.4 (direita):

$$g^*: \langle \Sigma_2^*, conc_2\rangle \to \langle \Sigma_1^*, conc_1\rangle$$

Sugere-se como exercício detalhar a definição indutiva do homomorfismo g^*. ❑

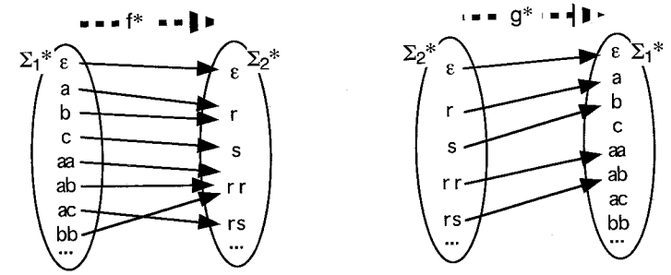


Figura 9.4 Homomorfismos de grupóides induzidos por funções entre alfabetos

Definição 9.10 - Isomorfismo de Grupóides
Um homomorfismo de grupóides $h: \langle A, \oplus\rangle \to \langle B, \otimes\rangle$ é dito um *Isomorfismo de Grupóides* se e somente se h possui um homomorfismo de grupóides inverso, ou seja, se existe um homomorfismo de grupóides $g: \langle B, \otimes\rangle \to \langle A, \oplus\rangle$ tal que:

$$g \circ h = id_{\langle A,\oplus\rangle} \quad \text{e} \quad h \circ g = id_{\langle B,\oplus\rangle}$$ ❑

EXEMPLO 9.17 - Isomorfismo de Grupóides
Seja $\Sigma = \{a\}$ um alfabeto. Considere os grupóides $\langle \Sigma^*, conc\rangle$ e $\langle \mathbf{N}, +\rangle$. Seja:

$$h: \Sigma \to \mathbf{N}$$

uma função tal que $h\langle a\rangle = 1$. Então $h: \Sigma \to \mathbf{N}$ induz o homomorfismo de grupóides:

$$h^*: \langle \Sigma^*, conc\rangle \to \langle \mathbf{N}, +\rangle$$

o qual é indutivamente definido como segue:

$h^*\langle\varepsilon\rangle = 0$
$h^*\langle a\rangle = h\langle a\rangle$
se $w \in \Sigma^*$, então $h^*\langle a\,w\rangle = h\langle a\rangle + h^*\langle w\rangle$

Observe que o homomorfismo apresentado mapeia cada palavra no seu correspondente tamanho (número de símbolos). Por exemplo:

$h^*\langle aaa\rangle =$
$h\langle a\rangle + h^*\langle aa\rangle =$
$h\langle a\rangle + h\langle a\rangle + h^*\langle a\rangle =$
$h\langle a\rangle + h\langle a\rangle + h\langle a\rangle + h^*\langle\varepsilon\rangle =$
$1 + 1 + 1 + 0 = 3$

De fato, o homomorfismo induzido $h^*: \langle \Sigma^*, conc \rangle \rightarrow \langle \mathbf{N}, + \rangle$ é um isomorfismo de grupóides (qual o homomorfismo inverso?). Logo, $\langle \Sigma^*, conc \rangle$ e $\langle \mathbf{N}, + \rangle$ são considerados, basicamente, o mesmo, ou seja, iguais a menos de isomorfismo. Nesse caso, como $\langle \mathbf{N}, + \rangle$ é abeliano, obrigatoriamente, $\langle \Sigma^*, conc \rangle$ também é abeliano. ❑

Observação 9.11 - Base de Sistemas Numéricos

Desde o ensino fundamental, a *base dos sistemas numéricos* usualmente adotada no estudo da Matemática é a *base decimal*, cuja origem está no número de dedos das duas mãos do ser humano. Trata-se de uma base especialmente complicada para representação em sistemas computadores. De fato, o computador usa a *base binária* (os dois valores possíveis de um *bit*: 0 e 1 ou qualquer outro alfabeto binário). Observe que, no EXEMPLO 9.17 - Isomorfismo de Grupóides, foi apresentada uma *base unária* para os números naturais e a correspondente operação de adição. O detalhamento de bases é desenvolvido em estudos como *Arquitetura de Computadores* e *Aritmética Computacional*. De forma análoga, sugere-se, como exercício, definir uma base binária para os números naturais. ❑

Os dois teoremas que seguem estabelecem importantes resultados, a saber:

- a composição de homomorfismos de grupóides é um homomorfismo de grupóides;
- homomorfismos de grupóides preservam a propriedade associativa.

Teorema 9.12 - Composição de Homomorfismo de Grupóides

Sejam $\langle A, \oplus \rangle$, $\langle B, \otimes \rangle$ e $\langle C, \nabla \rangle$ grupóides e $f: \langle A, \oplus \rangle \rightarrow \langle B, \otimes \rangle$ e $g: \langle B, \otimes \rangle \rightarrow \langle C, \nabla \rangle$ homomorfismos de grupóides. Então, o morfismo composto:

$$g \circ f: \langle A, \oplus \rangle \rightarrow \langle C, \nabla \rangle$$

induzido pela composição das funções $f: A \rightarrow B$ e $g: B \rightarrow C$, ou seja, pela função composta $g \circ f: A \rightarrow C$, é um homomorfismo de grupóides.

Prova:

Suponha que $f: \langle A, \oplus \rangle \rightarrow \langle B, \otimes \rangle$ e $g: \langle B, \otimes \rangle \rightarrow \langle C, \nabla \rangle$ são homomorfismos de grupóides. Como a composição de funções é uma função, para mostrar que $g \circ f: \langle A, \oplus \rangle \rightarrow \langle C, \nabla \rangle$ é um homomorfismo de grupóides, basta mostrar que a composição preserva a operação. Assim, para quaisquer $a_1 \in A$ e $a_2 \in A$, vale:

$g \circ f\langle a_1 \oplus a_2 \rangle =$	definição de composição
$g\langle f\langle a_1 \oplus a_2 \rangle\rangle =$	f é homomorfismo de grupóides
$g\langle f\langle a_1 \rangle \otimes f\langle a_2 \rangle\rangle =$	g é homomorfismo de grupóides
$g\langle f\langle a_1 \rangle\rangle \nabla g\langle f\langle a_2 \rangle\rangle =$	definição de composição
$g \circ f\langle a_1 \rangle \nabla g \circ f\langle a_2 \rangle$	

Portanto, $g \circ f\langle a_1 \oplus a_2 \rangle = g \circ f\langle a_1 \rangle \nabla g \circ f\langle a_2 \rangle$, ou seja, a operação é preservada pelo homomorfismo composto.

Logo, $g \circ f: \langle A, \oplus \rangle \rightarrow \langle C, \nabla \rangle$ é um homomorfismo de grupóides. ❑

O seguinte teorema mostra que, em um homomorfismo de grupóides, se um dos dois grupóides (origem ou destino) é semigrupo, então o homomorfismo também preserva a associatividade.

Teorema 9.13 - Homomorfismo de Grupóides Preserva a Associatividade

Sejam $\langle A, \oplus \rangle$ e $\langle B, \otimes \rangle$ grupóides e $f: \langle A, \oplus \rangle \rightarrow \langle B, \otimes \rangle$ um homomorfismo de grupóides. Se $\langle A, \oplus \rangle$ ou $\langle B, \otimes \rangle$ é semigrupo, então $f: \langle A, \oplus \rangle \rightarrow \langle B, \otimes \rangle$ preserva a associatividade.

Prova:

Caso 1: $\langle A, \oplus \rangle$ é semigrupo. Como $\oplus$ é associativa, para quaisquer $a \in A$, $b \in A$ e $c \in A$, vale:

$$f\langle a \oplus (b \oplus c) \rangle = f\langle (a \oplus b) \oplus c \rangle$$

Como $f: \langle A, \oplus \rangle \rightarrow \langle B, \otimes \rangle$ preserva a operação, vale:

$$f\langle a \oplus (b \oplus c) \rangle = f\langle (a \oplus b) \oplus c \rangle \Leftrightarrow$$
$$f\langle a \rangle \otimes f\langle (b \oplus c) \rangle = f\langle (a \oplus b) \rangle \otimes f\langle c \rangle \Leftrightarrow$$
$$f\langle a \rangle \otimes (f\langle b \rangle \otimes f\langle c \rangle) = (f\langle a \rangle \otimes f\langle b \rangle) \otimes f\langle c \rangle$$

Portanto, preserva a associatividade;

Caso 2: $\langle B, \otimes \rangle$ é semigrupo. Para quaisquer $a \in A$, $b \in A$ e $c \in A$, vale:

$f\langle a \oplus (b \oplus c) \rangle =$	f preserva a operação
$f\langle a \rangle \otimes f\langle b \oplus c \rangle =$	f preserva a operação
$f\langle a \rangle \otimes (f\langle b \rangle \otimes f\langle c \rangle) =$	$\otimes$ é associativa
$(f\langle a \rangle \otimes f\langle b \rangle) \otimes f\langle c \rangle =$	f preserva a operação
$f\langle a \oplus b \rangle \otimes f\langle c \rangle =$	f preserva a operação
$f\langle (a \oplus b) \oplus c \rangle$	

Portanto, preserva a associatividade. ❑

Portanto, um *homomorfismo de semigrupos* é como um homomorfismo de grupóides, no sentido em que basta preservar a operação.

Definição 9.14 - Homomorfismo de Semigrupos

Sejam $\langle A, \oplus \rangle$ e $\langle B, \otimes \rangle$ dois semigrupos. Um *Homomorfismo de Semigrupos*:

$$h: \langle A, \oplus \rangle \rightarrow \langle B, \otimes \rangle$$

é um homomorfismo de grupóides $h: \langle A, \oplus \rangle \rightarrow \langle B, \otimes \rangle$. ❑

Um homomorfismo de semigrupos $h: \langle A, \oplus \rangle \rightarrow \langle B, \otimes \rangle$ é dito um *isomorfismo de semigrupos* se e somente se h possui um homomorfismo de semigrupos inverso.

A preservação da comutatividade segue um raciocínio análogo ao da associatividade. Assim, a prova do seguinte teorema é sugerida como exercício.

Teorema 9.15 - Homomorfismo de Grupóides Preserva a Comutatividade

Sejam $\langle A, \oplus \rangle$ e $\langle B, \otimes \rangle$ grupóides e $f: \langle A, \oplus \rangle \rightarrow \langle B, \otimes \rangle$ um homomorfismo de grupóides. Se $\langle A, \oplus \rangle$ ou $\langle B, \otimes \rangle$ é abeliano, então $f: \langle A, \oplus \rangle \rightarrow \langle B, \otimes \rangle$ preserva a comutatividade. ❑

9.5.2 Homomorfismo de Monóides

Em se tratando de mapeamento de monóides, preservar a operação, não necessariamente implica preservar o elemento neutro (sugere-se como exercício comprovar tal fato). Assim, um homomorfismo de monóides é uma função entre os conjuntos suportes tal que, simultaneamente (veja a Figura 9.5):

- preserva a operação (como um homomorfismo de grupóides/semigrupos);
- preserva o elemento neutro.

Definição 9.16 - Homomorfismo de Monóides

Sejam $\langle A, \oplus, e_A \rangle$ e $\langle B, \otimes, e_B \rangle$ dois monóides. Um *Homomorfismo de Monóides*:

$$h: \langle A, \oplus, e_A \rangle \rightarrow \langle B, \otimes, e_B \rangle$$

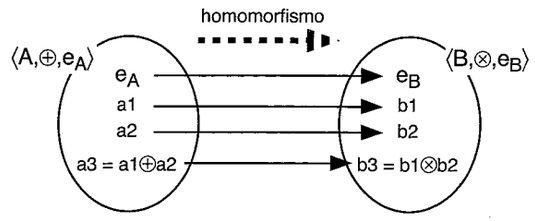


Figura 9.5 Homomorfismo de monóides

é um homomorfismo de semigrupos (ou de grupóides) $h: \langle A, \oplus\rangle \to \langle B, \otimes\rangle$ tal que preserva o elemento neutro, ou seja:

$$h\langle e_A\rangle = e_B$$ ❑

Analogamente aos grupóides (ou semigrupos), um homomorfismo de monóides é um *isomorfismo de monóides* se e somente se possuir um homomorfismo de monóides inverso.

EXEMPLO 9.18 - Homomorfismo: Identidade, Inclusão

Sejam $\langle A, \oplus, e\rangle$, $\langle \mathbf{N}, +, 0\rangle$ e $\langle \mathbf{Z}, +, 0\rangle$ monóides. Então:

a) *Identidade*. A função identidade $id_A: A \to A$ induz o homomorfismo identidade de monóides:

$$id_{\langle A,\oplus,e\rangle}: \langle A, \oplus, e\rangle \to \langle A, \oplus, e\rangle$$

Observe que o endomorfismo $id_{\langle A,\oplus,e\rangle}$ é um isomorfismo de monóides;

b) *Inclusão*. A função inclusão $inc_{\mathbf{N},\mathbf{Z}}: \mathbf{N} \to \mathbf{Z}$ induz o homomorfismo inclusão de monóides abelianos:

$$inc_{\langle \mathbf{N},+,0\rangle,\langle \mathbf{Z},+,0\rangle}: \langle \mathbf{N}, +, 0\rangle \to \langle \mathbf{Z}, +, 0\rangle$$ ❑

EXEMPLO 9.19 - Homomorfismo: União

Sejam $A = \{a, b\}$ e $X = \{x, y, z\}$ conjuntos. Considere os monóides abelianos $\langle \mathbf{P}(A), \cup, \varnothing\rangle$ e $\langle \mathbf{P}(B), \cup, \varnothing\rangle$, bem como o morfismo:

$$h: \langle \mathbf{P}(A), \cup, \varnothing\rangle \to \langle \mathbf{P}(X), \cup, \varnothing\rangle$$

tal que:

$$h\langle \varnothing\rangle = \varnothing$$
$$h\langle \{a\}\rangle = \{x, y\}$$
$$h\langle \{b\}\rangle = \{y, z\}$$
$$h\langle \{a, b\}\rangle = \{x, y, z\}$$

Então, h é um homomorfismo de monóides. De fato, além de preservar o elemento neutro, preserva a operação. Para exemplificar a preservação da operação (lembre-se de que um exemplo *não* é uma prova para *qualquer* caso), considere o seguinte:

$$h\langle \{a\} \cup \{b\}\rangle = h\langle \{a, b\}\rangle = \{x, y, z\} = \{x, y\} \cup \{y, z\} = h\langle \{a\}\rangle \cup h\langle \{b\}\rangle$$

e portanto, $h\langle \{a\} \cup \{b\}\rangle = h\langle \{a\}\rangle \cup h\langle \{b\}\rangle$

Para este exemplo, como seria uma prova de que, de fato, h sempre preserva a operação? ❑

EXEMPLO 9.20 - Homomorfismo: Isomorfismo

Seja $\Sigma = \{a\}$ um alfabeto. Considere os monóides $\langle \Sigma^*, conc, \varepsilon\rangle$ e $\langle \mathbf{N}, +, 0\rangle$. Então, o homomorfismo de grupóides apresentado no EXEMPLO 9.17 - Isomorfismo de Grupóides é um isomorfismo de monóides. De fato:

$$h^*: \langle \Sigma^*, conc, \varepsilon\rangle \to \langle \mathbf{N}, +, 0\rangle$$

além de preservar a operação (pois é um homomorfismo de grupóides), preserva o elemento neutro, ou seja:

$$h^*\langle \varepsilon\rangle = 0$$ ❑

O teorema que segue mostra que a composição de homomorfismos de monóides é um homomorfismo de monóides.

Teorema 9.17 - Composição de Homomorfismo de Monóides

Sejam $\langle A, \oplus, e_A\rangle$, $\langle B, \otimes, e_B\rangle$ e $\langle C, \nabla, e_C\rangle$ monóides e $f: \langle A, \oplus, e_A\rangle \to \langle B, \otimes, e_B\rangle$ e $g: \langle B, \otimes, e_B\rangle \to \langle C, \nabla, e_C\rangle$ homomorfismos de monóides. Então, o homomorfismo composto:

$$g \circ f: \langle A, \oplus, e_A\rangle \to \langle C, \nabla, e_C\rangle$$

induzido pela composição das funções $f: A \to B$ e $g: B \to C$, é um homomorfismo de monóides.

Prova: (*direta*)

Suponha que $f: \langle A, \oplus, e_A\rangle \to \langle B, \otimes, e_B\rangle$ e $g: \langle B, \otimes, e_B\rangle \to \langle C, \nabla, e_C\rangle$ são homomorfismos de monóides. Como a composição de dois homomorfismos de semigrupos é um homomorfismo de semigrupos, para mostrar que $g \circ f: \langle A, \oplus, e_A\rangle \to \langle C, \nabla, e_C\rangle$ é um homomorfismo de monóides, basta mostrar que a composição preserva o elemento neutro. De fato:

$g \circ f\langle e_A\rangle =$	definição de composição
$g\langle f\langle e_A\rangle\rangle =$	f é homomorfismo de monóides
$g\langle e_B\rangle =$	g é homomorfismo de monóides
e_C	

Portanto, $g \circ f\langle e_A\rangle = e_C$, ou seja, o elemento neutro é preservado pelo homomorfismo composto.

Logo, $g \circ f: \langle A, \oplus, e_A\rangle \to \langle C, \nabla, e_C\rangle$ é um homomorfismo de monóides. ❑

9.5.3 Homomorfismo de Grupos

Um *homomorfismo de grupos* é como um homomorfismo de grupóides, no sentido em que basta preservar a operação. De fato, a seguir, prova-se que, em se tratando de grupos, ao preservar a operação, automaticamente, preserva-se o elemento neutro e o elemento inverso. Adicionalmente, como a composição de homomorfismos de grupóides é um homomorfismo de grupóides, a composição de homomorfismos de grupos é um homomorfismo de grupos.

Por fim (e analogamente aos casos anteriores), um homomorfismo de grupos é um *isomorfismo de grupos* se e somente se possuir um homomorfismo de grupos inverso.

Na prova do teorema que segue, é usado o resultado do Teorema 9.6 - Propriedade de Cancelamento dos Grupos.

Teorema 9.18 - Homomorfismo de Grupos × Elemento Neutro

Sejam $\langle A, \oplus, e_A\rangle$ e $\langle B, \otimes, e_B\rangle$ grupos e $f: \langle A, \oplus, e_A\rangle \to \langle B, \otimes, e_B\rangle$ um homomorfismo de grupóides. Então, f preserva o elemento neutro, ou seja:

$$f\langle e_A\rangle = e_B$$

Prova:

Suponha $\langle A, \oplus, e_A\rangle$ e $\langle B, \otimes, e_B\rangle$ grupos e $f: \langle A, \oplus, e_A\rangle \to \langle B, \otimes, e_B\rangle$ um homomorfismo de grupóides. Então:

$f\langle e_A\rangle \otimes f\langle e_A\rangle =$	f é homomorfismo de grupóides
$f\langle e_A \oplus e_A\rangle =$	elemento neutro de ⊕

$f\langle e_A\rangle =$ elemento neutro de $\otimes$
$f\langle e_A\rangle \otimes e_B$

Portanto, $f\langle e_A\rangle \otimes f\langle e_A\rangle = f\langle e_A\rangle \otimes e_B$. Pela propriedade do cancelamento (à esquerda), vale:

$$f\langle e_A\rangle = e_B$$

Logo, f preserva o elemento neutro. ❑

Na prova do teorema que segue, são usados os resultados dos seguintes teoremas:

- Teorema 9.18 - Homomorfismo de Grupos x Elemento Neutro;
- Teorema 9.6 - Propriedade de Cancelamento dos Grupos.

Teorema 9.19 - Homomorfismo de Grupos x Elemento Inverso
Sejam $\langle A, \oplus, e_A\rangle$ e $\langle B, \otimes, e_B\rangle$ grupos e $f: \langle A, \oplus, e_A\rangle \rightarrow \langle B, \otimes, e_B\rangle$ um homomorfismo de grupóides. Então f preserva o elemento inverso, ou seja:

$$(\forall a \in A)\,(f\langle \underline{a}\rangle = \underline{f\langle a\rangle})$$

Prova:
Suponha $\langle A, \oplus, e_A\rangle$ e $\langle B, \otimes, e_B\rangle$ grupos e $f: \langle A, \oplus, e_A\rangle \rightarrow \langle B, \otimes, e_B\rangle$ um homomorfismo de grupóides. Para qualquer $a \in A$, vale:

$f\langle \underline{a}\rangle \otimes f\langle a\rangle =$ f é homomorfismo de grupóides
$f\langle \underline{a} \oplus a\rangle =$ elemento inverso de $\oplus$
$f\langle e_A\rangle =$ f preserva elemento neutro
$e_B =$ elemento inverso de $\otimes$
$\underline{f\langle a\rangle} \otimes f\langle a\rangle$ f é homomorfismo de grupóides

Portanto, $f\langle \underline{a}\rangle \otimes f\langle a\rangle = \underline{f\langle a\rangle} \otimes f\langle a\rangle$. Pela propriedade do cancelamento (à direita), vale:

$$f\langle \underline{a}\rangle = \underline{f\langle a\rangle}$$

Logo, f preserva o elemento inverso. ❑

9.6 Monóide Livre Gerado e Fecho de Kleene

Já foi visto que:

a) Para um dado alfabeto Σ, a álgebra interna $\langle \Sigma^*, conc, \varepsilon\rangle$ é um monóide;
b) O conjunto Σ^* pode ser indutivamente definido a partir do alfabeto Σ, usando a operação de concatenação, como segue:
   b.1) *Base de Indução.*
   $\varepsilon \in \Sigma^*$
   para qualquer $x \in \Sigma$, vale $x \in \Sigma^*$
   b.2) *Passo de Indução.*
   se u e v são palavras de Σ^*, então a concatenação u v é palavra de Σ^*

Observe que o mesmo raciocínio pode ser aplicado sobre um conjunto A qualquer, o qual não necessariamente é um alfabeto (ou seja, não necessariamente é finito). Adicionalmente, no passo de indução, pode-se considerar a possibilidade da palavra vazia, dado que ε é o elemento neutro da concatenação, ou seja:

$$\text{se } u \in A^*, \text{ então } u\,\varepsilon = \varepsilon\,u = u$$

Tal construção resulta em um tipo especial e importante de monóide denominado de *monóide livre*, no qual o conjunto suporte A^* é (livremente) gerado por um conjunto qualquer A.

Definição 9.20 - Monóide Livre, Fecho de Kleene
Seja A um conjunto. Um *Monóide Livre Gerado por* A ou *Monóide Livremente Gerado por* A é a álgebra interna:

$$\langle A^*, conc, \varepsilon\rangle$$

em que $conc: A^* \times A^* \rightarrow A^*$ é a operação de concatenação, a palavra vazia ε é o elemento neutro, e o conjunto suporte A^*, também denominado de *Fecho de Kleene de* A, é indutivamente definido como segue:

a) *Base de Indução.*
   $\varepsilon \in A^*$
   para qualquer $x \in A$, vale $x \in A^*$
b) *Passo de Indução.*
   se u e v são palavras de A^*, então a concatenação u v é palavra de A^*

Nesse contexto, o conjunto A é denominado de *Gerador.* ❑

Portanto, para um dado alfabeto, vale:

- o conjunto de todas as palavras definidas sobre um alfabeto é o conjunto suporte do monóide livremente gerado pelo alfabeto, ou seja, o Fecho de Kleene do alfabeto;
- uma *linguagem formal* (como, por exemplo, Pascal) é simplesmente um subconjunto do Fecho de Kleene do alfabeto.

Em que condições um monóide livre gerado por um conjunto é um monóide abeliano?

EXEMPLO 9.21 - Homomorfismo: Monóides Livres
Sejam $\Sigma_1 = \{a, b, c\}$ e $\Sigma_2 = \{r, s\}$ alfabetos. Considere os monóides $\langle \Sigma_1^*, conc_1, \varepsilon\rangle$ e $\langle \Sigma_2^*, conc_2, \varepsilon\rangle$. Então:

a) Considere o EXEMPLO 9.16 - Homomorfismo de Grupóides: Concatenação. Observe que os homomorfismos definidos são homomorfismos de monóides livres (preservam a operação e o elemento neutro). Os dois homomorfismos apresentados são (canonicamente) induzidos por uma função entre os conjuntos geradores. De fato, tal tipo de homomorfismo (induzido por uma função entre os geradores) é usual;
b) Um homomorfismo de monóides livres $h: \langle \Sigma_1^*, conc_1, \varepsilon\rangle \rightarrow \langle \Sigma_2^*, conc_2, \varepsilon\rangle$ *não* induzido por uma função entre os conjuntos geradores é indutivamente definido como segue:

$h\langle \varepsilon\rangle = \varepsilon$
$h\langle a\rangle = rr$
$h\langle b\rangle = rs$
$h\langle c\rangle = ss$
se $x \in \Sigma_1$ e $w \in \Sigma_1^*$, então $h\langle x\,w\rangle = h\langle x\rangle\, h\langle w\rangle$

Por exemplo:

$h\langle abc\rangle =$
$h\langle a\rangle\, h\langle bc\rangle =$
$h\langle a\rangle\, h\langle b\rangle\, h\langle c\,\varepsilon\rangle =$
$h\langle a\rangle\, h\langle b\rangle\, h\langle c\rangle\, h\langle \varepsilon\rangle =$
$rr\,rs\,ss\,\varepsilon =$
$rr\,rs\,ss$ ❑

9.7 Grafos

Até o momento, todas as álgebras apresentadas consideram uma única operação a qual é definida sobre um único conjunto. Entretanto, freqüentemente, as álgebras:

- possuem mais de uma operação;
- as operações são definidas sobre mais de um conjunto. Neste caso, são denominadas de *álgebras polissortidas* (do termo *sorte*, no sentido de gênero, classe ou espécie). De forma análoga, as álgebras definidas sobre um único conjunto são denominadas de *álgebras monossortidas* .

A noção de grafo já foi informalmente apresentada quando do estudo das endorrelações. De fato, já foi visto que toda endorrelação pode ser vista como um grafo (e conseqüentemente, como uma álgebra). Entretanto, o inverso não é verdadeiro: nem todo grafo pode ser visto como uma endorrelação (a verificação de tal resultado é sugerida como exercício).

Grafos são especialmente importantes para Computação e Informática e seu estudo é apenas brevemente apresentado neste livro. A ênfase do que segue é sobre grafos pequenos (nodos e arcos constituem conjuntos) e direcionados (arcos possuem sentido). Um grafo pode ser visto como uma álgebra como segue:

- *nodos* e *arcos* são conjuntos sobre os quais as operações são definidas. Portanto, trata-se de uma álgebra polissortida;
- *origem* e *destino* são duas operações unárias fechadas (não-internas) as quais associam, para cada arco, o seu nodo origem e destino, respectivamente.

Definição 9.21 - Grafo

Um *Grafo Pequeno Direcionado* ou simplesmente *Grafo* G é uma álgebra polissortida:

$$G = \langle V, T, orig, dest \rangle$$

na qual:

a) V é um conjunto de *Nodos* ou *Vértices*;
b) T é um conjunto de *Arcos*, *Arestas* ou *Setas*;
c) $orig: T \to V$ e $dest: T \to V$ são operações totais (funções) denominadas *Origem* e *Destino*, respectivamente. ❑

Um arco t tal que $orig\langle t \rangle = A$ e $dest\langle t \rangle = B$ é normalmente denotado por $t: A \to B$. É usual representar grafos na forma de diagramas (quando possível) nos quais cada nodo é representado por um círculo ou ponto, e cada arco, por uma seta, respeitando seus nodos origem e destino. Por exemplo, o arco $t: A \to B$ é ilustrado na Figura 9.6 (esquerda).

EXEMPLO 9.22 - Grafo

a) *Arco único.* $G_1 = \langle \{A, B\}, \{t\}, orig_1, dest_1 \rangle$, ilustrado na Figura 9.6, esquerda, onde:

$$orig\langle t \rangle = A$$
$$dest\langle t \rangle = B$$

b) *Nodo isolado.* $G_2 = \langle \{X\}, \varnothing, orig_2, dest_2 \rangle$, onde $orig_2: \varnothing \to \{X\}$ e $dest_2: \varnothing \to \{X\}$ são funções vazias, como ilustrado na Figura 9.6, centro;
c) *Arcos paralelos.* $G_3 = \langle \{1, 2\}, \{r, s, t, u, v\}, \partial_{03}, \partial_{13} \rangle$, como ilustrado na Figura 9.6, direita. Dois arcos são ditos paralelos se possuem os mesmos nodos origem e destino. Note-se que r, s e t são arcos paralelos, assim como u e v. ❑

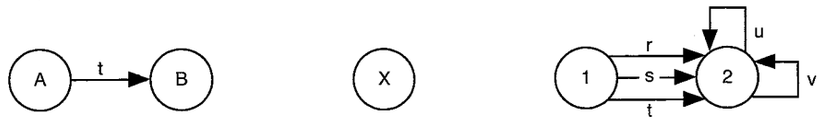


Figura 9.6 Grafos

Muitas publicações consideram que um grafo não pode possuir arcos paralelos. Se tal ocorrer, a estrutura é considerada um *multigrafo*.

Analogamente às álgebras monossortidas apresentadas, um homomorfismo de grafos deve preservar a estrutura dos grafos, ou seja, ao mapear os nodos e arcos, deve preservar as operações de origem e de destino. Portanto, o mapeamento de um arco deve ser de acordo com o mapeamento dos seus correspondentes nodos origem e destino.

Definição 9.22 - Homomorfismo de Grafos

Sejam $G_1 = \langle V_1, T_1, orig_1, dest_1 \rangle$ e $G_2 = \langle V_2, T_2, orig_2, dest_2 \rangle$ grafos. Um *Homomorfismo de Grafos*:

$$h: G_1 \to G_2$$

é um par de funções:

$$h = \langle h_V, h_T \rangle$$

onde:

$$h_V: V_1 \to V_2 \quad e \quad h_T: T_1 \to T_2$$

são tais que (sendo "o" a composição de funções):

$$orig_2 \circ h_T = h_V \circ orig_1 \quad e \quad dest_2 \circ h_T = h_V \circ dest_1$$ ❑

EXEMPLO 9.23 - Homomorfismo de Grafos

Suponha os grafos:

$$G_1 = \langle \{A, B, C\}, \{r, s, t\}, orig_1, dest_1 \rangle$$
$$G_2 = \langle \{1, 2, 3\}, \{a, b, c, d, e\}, orig_2, dest_2 \rangle$$

ilustrados na Figura 9.7. Note-se que o homomorfismo de grafos:

$$h = \langle h_V, h_T \rangle: G_1 \to G_2$$

ilustrado na mesma figura, preserva origem e destino dos arcos (existe mais algum homomorfismo de grafos de G_1 para G_2? Existe pelo menos um morfismo de G_2 para G_1? Por quê?). ❑

Teorema 9.23 - Composição de Homomorfismos de Grafos

Sejam $G_1 = \langle V_1, T_1, orig_1, dest_1 \rangle$, $G_2 = \langle V_2, T_2, orig_2, dest_2 \rangle$ e $G_3 = \langle V_3, T_3, orig_3, dest_3 \rangle$ grafos e $f: G_1 \to G_2$ e $g: G_2 \to G_3$ homomorfismos de grafos. Então, o homomorfismo composto:

$$g \circ f: G_1 \to G_3$$

tal que $g \circ f = \langle g_V \circ f_V, g_T \circ f_T \rangle$, é um homomorfismo de grafos.

<u>Prova:</u> *(direta)*

Seja $t: A \to B$ arco do grafo G_1. Então:

$t: A \to B$ é arco de G_1 $\Rightarrow$
$f_T(t): f_V(A) \to f_V(B)$ é arco de G_2, pois f é homomorfismo de grafos $\Rightarrow$
$g_T(f_T(t)): g_V(f_V(A)) \to g_V(f_V(B))$ é arco de G_3, pois g é homomorfismo de grafos

Logo, $g \circ f$ é homomorfismo de grafos. ❑

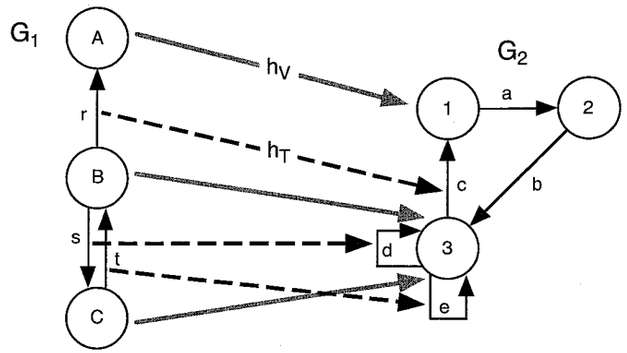


Figura 9.7 Homomorfismo de grafos preserva origem e destino

9.8 Categorias

Em um primeiro momento, a *Teoria das Categorias* pode ser vista como uma generalização da álgebra de funções. Nesse contexto, claramente, a principal operação sobre funções é a de composição. Na realidade, uma *Categoria* é uma estrutura abstrata, constituída de *objetos* e *setas* entre os objetos, com uma propriedade fundamental que é a *composicionalidade das setas*. Por exemplo, objetos e setas podem ser conjuntos e funções. Alguns exemplos de categorias são ilustrados na Figura 9.8.

Em uma categoria, qualquer modificação sobre os objetos, as setas ou a composição resulta em uma nova categoria. Por exemplo, na primeira categoria da Figura 9.8, a substituição das funções (totais) por funções parciais resulta em uma nova categoria (conjuntos e funções parciais), com diferentes propriedades. Como ilustração, nesse caso, um simples produto cartesiano (generalizado categorialmente) de dois conjuntos é diferente nas duas categorias.

É importante observar que as noções de objeto, seta e composição não necessariamente possuem estruturas que lembrem as usadas na Teoria dos Conjuntos. Por exemplo, no caso de um conjunto parcialmente ordenado visto como uma categoria, os objetos não possuem qualquer estrutura (um objeto é um elemento de um conjunto), as setas são pares de elementos, e a composição é dada pela transitividade da relação.

Adicionalmente, uma mesma estrutura pode constituir uma categoria por si só, ou ser objeto de uma categoria, como nos dois casos referentes aos conjuntos parcialmente ordenados ilustrados na Figura 9.8 (*funções monotônicas*, ou seja, funções que preservam a estrutura de ordem dos conjuntos, são formalmente apresentadas em capítulo subseqüente). Assim, a categoria dos conjuntos parcialmente ordenados pode ser vista como uma categoria de categorias (ou seja, uma categoria cujos objetos são conjuntos parcialmente ordenados vistos como categorias).

Teoria das Categorias e *Ciência da Computação* não só possuem muito em comum como são enriquecidas mutuamente a partir de visões e abordagens de um campo sobre o outro. Entre as diversas características da Teoria das Categorias que motivam o seu uso na Computação, destaca-se a expressividade de suas construções conforme é explicitamente destacado nas Diretrizes Curriculares do MEC para Cursos de Computação e Informática, como segue:

Categoria	Objetos	Setas	Composição
Conjuntos e Funções	conjuntos	funções (totais)	composição de funções
Figuras	figuras	transformações de figuras	construtor de transformações "complexas"
Um Programa Funcional	tipos de dados	operações	construtor de operações não-primitivas
Espaços Vetoriais	espaços vetoriais	transformações lineares	composição de transformações lineares
Grafos	grafos	homomorfismo de grafos	composição de homomorfismo de grafos
Lógica	proposições	provas	transitividade das provas
Uma Máquina de Estados	estados	transições	construtor de computações
Conjuntos Parcialmente Ordenados	conjuntos parcialmente ordenados	funções monotônicas	composição de funções monotônicas
Um Conjunto Parcialmente Ordenado	elementos do conjunto	pares da relação de ordem parcial	transitividade da relação de ordem parcial

Figura 9.8 Exemplos de categorias

Teoria das Categorias possui construções cujo poder de expressão não possui, em geral, paralelo em outras teorias. Esta expressividade permite formalizar idéias mais complexas de forma mais simples, bem como propicia um novo ou melhor entendimento das questões relacionadas com toda a Ciência da Computação. Como Teoria das Categorias é uma ferramenta nova, para exemplificar, vale a pena estabelecer um paralelo com a linguagem Pascal: Teoria das Categorias está para a Teoria dos Conjuntos assim como Pascal está para a linguagens Assembler.

No que segue, apenas o conceito de categoria (como álgebra) é apresentado. O estudo de suas propriedades e aplicações não é objetivo desta publicação. Uma categoria é basicamente um grafo (eventualmente grande) no qual os arcos são componíveis, formando caminhos, e cada nodo possui um endoarco especial, com um significado de identidade.

Definição 9.24 - Categoria

Uma *Categoria* $\mathbf{C}$ é uma álgebra polissortida:

$$\mathbf{C} = \langle Ob_{\mathbf{C}}, Mor_{\mathbf{C}}, orig, dest, \iota, \circ \rangle$$

onde:

a) $Ob_{\mathbf{C}}$ é uma coleção de *Objetos*;

b) $Mor_{\mathbf{C}}$ é uma coleção de *Morfismos* ou *Setas*;

c) $orig: Mor_{\mathbf{C}} \rightarrow Ob_{\mathbf{C}}$ e $dest: Mor_{\mathbf{C}} \rightarrow Ob_{\mathbf{C}}$ são operações fechadas e totais denominadas *Origem* e *Destino*, respectivamente. Um morfismo f de $Mor_{\mathbf{C}}$ tal que $orig(f) = A$ e $dest(f) = B$ é usualmente denotado por:

$$f: A \rightarrow B$$

d) $\circ: (\mathrm{Mor}_{\mathbf{C}})^2 \to \mathrm{Mor}_{\mathbf{C}}$ é uma operação denominada *Composição* tal que cada par de $\mathrm{Mor}_{\mathbf{C}}{}^2$:

$$\langle f: A \to B, g: B \to C \rangle$$

é associado a um morfismo:

$$g \circ f: A \to C$$

A operação de composição deve satisfazer a *propriedade associativa*, segundo a qual, para quaisquer morfismos $f: A \to B$, $g: B \to C$ e $h: C \to D$ de $\mathrm{Mor}_{\mathbf{C}}$, vale:

$$(h \circ g) \circ f = h \circ (g \circ f)$$

e) $\iota: \mathrm{Ob}_{\mathbf{C}} \to \mathrm{Mor}_{\mathbf{C}}$ é uma operação denominada *Identidade* tal que cada objeto A de $\mathrm{Ob}_{\mathbf{C}}$ é associado a um morfismo freqüentemente denotado como segue:

$$\iota_A: A \to A$$

A operação identidade deve satisfazer a *propriedade da identidade*, segundo a qual, para qualquer morfismo $f: A \to B$ em $\mathrm{Mor}_{\mathbf{C}}$, vale:

$$f \circ \iota_A = \iota_B \circ f = f$$

❑

EXEMPLO 9.24 - Categoria **Set**

A categoria **Set** é tal que possui todos os conjuntos como objetos, todas as funções (totais) como morfismos, e as operações categoriais de composição e identidade são dadas pela composição de funções e pela função identidade de cada conjunto, respectivamente. Ou seja:

$$\mathbf{Set} = \langle \mathrm{Ob}_{\mathbf{Set}}, \mathrm{Mor}_{\mathbf{Set}}, \mathrm{orig}, \mathrm{dest}, \iota, \circ \rangle$$

a) $\mathrm{Ob}_{\mathbf{Set}}$ é a coleção de todos os conjuntos;
b) $\mathrm{Mor}_{\mathbf{Set}}$ é a coleção de todas as funções (totais);
c) $\mathrm{orig}: \mathrm{Mor}_{\mathbf{Set}} \to \mathrm{Ob}_{\mathbf{Set}}$ e $\mathrm{dest}: \mathrm{Mor}_{\mathbf{Set}} \to \mathrm{Ob}_{\mathbf{Set}}$ são tais que, para qualquer função f com domínio em A e codomínio em B, vale $\mathrm{orig}(f) = A$ e $\mathrm{dest}(f) = B$
d) $\circ: (\mathrm{Mor}_{\mathbf{Set}})^2 \to \mathrm{Mor}_{\mathbf{Set}}$ é a operação de composição de funções a qual, como já foi verificado, é associativa;
e) $\iota: \mathrm{Ob}_{\mathbf{Set}} \to \mathrm{Mor}_{\mathbf{Set}}$ é a operação identidade, segundo a qual cada conjunto A é associado à função identidade $\mathrm{id}_A: A \to A$, ou seja:

$$\iota\langle A \rangle = \mathrm{id}_A$$

❑

EXEMPLO 9.25 - Categoria **Mon**

A categoria **Mon** é tal que possui todos os monóides como objetos, todos os homomorfismos de monóides como morfismos, e as operações categoriais de composição e identidade são dadas pela composição de homomorfismos de monóides e pelo homomorfismo identidade de cada monóide, respectivamente. De forma análoga, pode-se definir as categorias constituídas por:

- grupóides e correspondentes homomorfismos;
- semigrupos e corrrespondentes homomorfismos;
- grupos e corrrespondentes homomorfismos. ❑

EXEMPLO 9.26 - Categoria **Gr**

A categoria **Gr** é tal que possui todos os grafos como objetos, todos os homomorfismos de grafos como morfismos, e as operações categoriais de composição e identidade são dadas pela composição de homomorfismos de grafos e pelo homomorfismo identidade de cada grafo, respectivamente. ❑

EXEMPLO 9.27 - As "Menores" Categorias

As "menores" categorias em termos do número de objetos e de morfismos são as seguintes:

a) *Categoria Vazia.* A menor categoria é a categoria que não possui objetos nem morfismos. Formalmente, é dada pela seguinte álgebra (procure justificar cada componente):

$$\langle \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing \rangle$$

b) *Categoria* **1**. A categoria com somente um objeto e um morfismo (no caso, o morfismo identidade desse objeto) é ilustrada na Figura 9.9 (esquerda);
c) *Categoria* **1+1**. A categoria com dois objetos e dois morfismos (identidade) é ilustrada na Figura 9.9 (centro);
d) *Categoria* **2**. A categoria com dois objetos e três morfismos (sendo dois identidade) é ilustrada na Figura 9.9 (direita). ❑

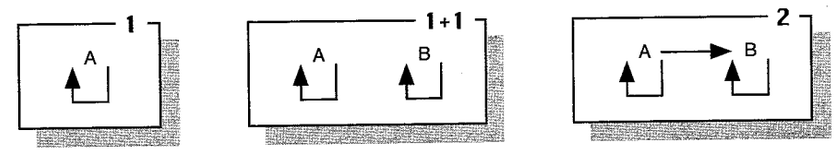


Figura 9.9 As "menores" categorias

No exemplo acima, é importante observar que, em qualquer das categorias, os objetos e os morfismos *não* necessariamente são conjuntos, nem funções, respectivamente. Com o objetivo de reforçar essa idéia, o exemplo a seguir introduz uma "visão categorial" da história conhecida como "os três porquinhos e o lobo mau".

EXEMPLO 9.28 - Categoria (Três Porquinhos e o Lobo Mau)

A história "os três porquinhos e o lobo mau" pode ser interpretada como: os três porquinhos P_1, P_2 e P_3, o lobo mau L e os morfismos com origem no lobo mau e destino em cada um dos porquinhos (o lobo persegue cada um dos porquinhos), além dos morfismos identidade (interpretados como "outra atividade qualquer que não envolve perseguição"). Tal situação constitui uma categoria a qual é ilustrada na Figura 9.10. ❑

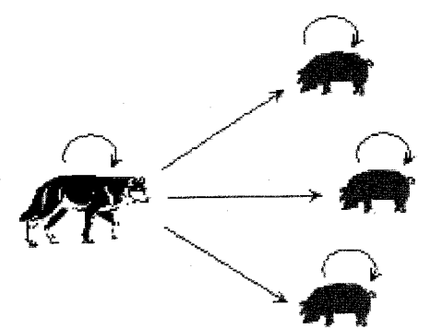


Figura 9.10 Categoria "os três porquinhos e o lobo mau"

9.9 Exercícios

Exercício 9.1 Considere o EXEMPLO 9.1 - Operação. Para quais operações exemplificadas faz sentido verificar se satisfaz as propriedades comutativa, associativa, de elemento neutro e de elemento inverso?

Exercício 9.2 Considere a Observação 9.3 - Elemento Neutro à Esquerda e à Direita. Apresente uma outra operação (diferente da divisão) que satisfaça a propriedade de elemento neutro ou à esquerda ou à direita, mas *não* ambas.

Exercício 9.3 Seja $A = \{a, b, c\}$. Considere a operação interna $\oplus: A^2 \rightarrow A$ definida pela tabela ilustrada na Figura 9.11. Verifique e justifique se a operação satisfaz cada uma das seguintes propriedades:

a) Fechada;
b) Associativa;
c) Elemento neutro;
d) Elemento inverso;
e) Comutativa.

$\oplus$	a	b	c
a	a	a	a
b	a	b	a
c	a	a	c

Figura 9.11 Operação interna

Exercício 9.4 Considere uma determinada operação interna $\oplus: A^2 \rightarrow A$ e a correspondente álgebra $\langle A, \oplus \rangle$. Freqüentemente é conveniente representar a operação na forma de tabela, também denominada de *tábua*, analogamente ao caso ilustrado na Figura 9.11. Neste caso, uma série de propriedades pode ser inferida simplesmente analisando a tábua. Para cada uma das seguintes propriedades, qual a correspondente análise da tábua que deve ser realizada?

a) Fechada;
b) Elemento neutro;
c) Elemento inverso;
d) Comutativa.

No caso específico do elemento neutro (analisando a tábua):

e) Procure concluir a razão da sua unicidade.

Dica: suponha que o elemento neutro não é único e analise a tábua, considerando que o elemento deve ser neutro à esquerda e à direita.

Exercício 9.5 Seja Σ um alfabeto. Considere o grupóide $\langle \Sigma^*, \text{conc} \rangle$, sendo que a operação de concatenação é tal que:

$$\text{conc}: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$$

Para cada um dos seguintes casos, verifique qual o correspondente tipo da álgebra interna, bem como se é abeliana (justifique a sua resposta):

a) Σ é um alfabeto vazio;
b) Σ é um alfabeto unitário;

c) Σ é um alfabeto tal que $\#\Sigma \geq 2$.

Exercício 9.6 A operação de união definida sobre todos os conjuntos constitui um grupóide? Analogamente para a operação de intersecção?

Dica: a classe de todos os conjuntos constitui um conjunto?

Exercício 9.7 Verifique se as seguintes álgebras internas são grupóides. Justifique a sua resposta:

a) Subtração nos seguintes casos:

$$\langle \mathbf{N}, - \rangle \quad \langle \mathbf{Z}, - \rangle \quad \langle \mathbf{R}, - \rangle$$

b) Divisão nos seguintes casos:

$$\langle \mathbf{R}, / \rangle \quad \langle \mathbf{R} - \{0\}, / \rangle \quad \langle \mathbf{N} - \{0\}, \text{DIV} \rangle$$

sendo DIV a divisão inteira (por exemplo, $7 \text{ DIV } 2 = 3$)

c) Produto cartesiano no conjunto das partes $\langle \mathbf{P}(A), \times \rangle$ nos seguintes casos:

A é tal que $\#A = 0$
A é tal que $\#A = 1$
A é tal que $\#A > 1$

Exercício 9.8 Justifique por que cada uma das seguintes álgebras *não* é um semigrupo:

a) Subtração nos inteiros: $\langle \mathbf{Z}, - \rangle$
b) Divisão nos reais sem o zero: $\langle \mathbf{R} - \{0\}, / \rangle$

Exercício 9.9 Considere a operação $\Diamond$ em $\mathbf{R}$ definida por:

$$x \Diamond y = a x + b y + c x y$$

onde a, b e c são números reais dados. Determine as condições para a, b e c de modo que $\langle \mathbf{R}, \Diamond \rangle$ constitua um monóide.

Exercício 9.10 Relativamente ao um monóide cujo conjunto suporte é unitário $\langle \{*\}, ! \rangle$ sendo que a operação $!: \{*\} \times \{*\} \rightarrow \{*\}$ é tal que $* ! * = *$, responda:

a) O monóide é abeliano?
b) Por que este é o menor monóide em termos de número de elementos do conjunto suporte?

Exercício 9.11 Justifique por que são grupos abelianos as seguintes álgebras internas:

a) $\langle \mathbf{Z}, + \rangle$
b) $\langle \mathbf{R}, + \rangle$
c) $\langle \mathbf{R} - \{0\}, * \rangle$
d) $\langle \{*\}, ! \rangle$ (ver exercício acima)

Exercício 9.12 Suponha A um conjunto não-vazio. Justifique por que não são grupos as seguintes álgebras internas:

a) $\langle \mathbf{N}, + \rangle$
b) $\langle \mathbf{R}, * \rangle$
c) $\langle \mathbf{P}(A), \cup \rangle$
d) $\langle \mathbf{P}(A), \cap \rangle$

Exercício 9.13 Considere a álgebra interna $\langle \mathbf{R}, \oplus \rangle$, sendo que a operação $\oplus$ é definida por:

$$x \oplus y = x - y + 3$$

Mostre que $\langle \mathbf{R}, \oplus \rangle$ *não* é um grupo comutativo.

Exercício 9.14 Já foi afirmado que a *Álgebra de Funções*, constituída por todas as funções e a operação de composição de funções, é uma álgebra grande e, portanto, não é um grupóide. Entretanto, observe que as álgebras de funções abaixo são pequenas. Para cada caso, verifique o correspondente tipo de álgebra de funções:

a) Álgebra constituída por todas as endofunções sobre $\varnothing$;
b) Álgebra constituída por todas as endofunções sobre um conjunto finito e não-vazio A;
c) Álgebra constituída por todas as funções definidas sobre dois conjuntos finitos A e B.

Exercício 9.15 A tabela-verdade ilustrada na Figura 9.12 apresenta as 16 possíveis operações binárias internas e fechadas (conetivos) sobre o conjunto $\{F, V\}$. Algumas dessas operações são especialmente importantes, como:

b *conjunção*
h *disjunção*
n *condição*
j *bicondição*
p *tautologia*
a *contradição*
g EXOR
o NAND
i NOR (abreviatura dos termos em inglês *not* e *or*)

Na última linha da tabela, o símbolo ✔ indica operações associativas. Claramente, todas as operações da tabela constituem um grupóide sobre $\{F, V\}$. Verifique e justifique:

a) Quais são semigrupos?
b) Quais são monóides?
c) Quais são grupos?

Operandos		16 possíveis operações binárias internas															
X	Y	a	b	c	d	e	f	g	h	i	j	k	L	m	n	o	p
V	V	F	V	F	V	F	V	F	V	F	V	F	V	F	V	F	V
V	F	F	F	V	V	F	F	V	V	F	F	V	V	F	F	V	V
F	V	F	F	F	F	V	V	V	V	F	F	F	F	V	V	V	V
F	F	F	F	F	F	F	F	F	F	V	V	V	V	V	V	V	V
Assoc.?		✔	✔		✔		✔	✔	✔		✔						✔

Figura 9.12 Conetivos possíveis

Exercício 9.16 Considere o Teorema 9.6 - Propriedade de Cancelamento dos Grupos. Desenvolva a prova referente ao cancelamento à esquerda.

Exercício 9.17 Considere o Teorema 9.7 - Elemento Inverso em um Grupo é Único. Desenvolva a correspondente prova.

Dica: use a propriedade de cancelamento dos grupos.

Exercício 9.18 Para cada um dos seguintes casos, verifique se existe e, neste caso, detalhe como seria um homomorfismo de grupóides, dado que:

a) O conjunto suporte do grupóide origem é vazio;
b) O conjunto suporte do grupóide destino é vazio;
c) O conjunto suporte do grupóide origem é unitário;
d) O conjunto suporte do grupóide destino é unitário.

Exercício 9.19 Analogamente ao EXEMPLO 9.15 - Morfismo Não-Homomorfismo de Grupóides, apresente um caso em que uma função inclusão (não-identidade) induz um morfismo o qual não é homomorfismo de grupóides.

Exercício 9.20 Suponha os alfabetos $\Sigma_1 = \{a, b, c\}$ e $\Sigma_2 = \{r, s\}$. Seja $g: \Sigma_2 \to \Sigma_1$ uma função tal que:

$$g\langle r\rangle = a \quad \text{e} \quad g\langle s\rangle = b$$

Detalhe a definição indutiva do seguinte homomorfismo de grupóides induzido pela função g:

$$g^*: \langle \Sigma_2^*, conc_2\rangle \to \langle \Sigma_1^*, conc_1\rangle$$

Exercício 9.21 Relativamente ao EXEMPLO 9.17 - Isomorfismo de Grupóides:

a) Mostre que o homomorfismo $h^*: \langle \Sigma^*, conc\rangle \to \langle \mathbf{N}, +\rangle$ possui inverso.
b) Defina uma base binária (na forma de um isomorfismo de grupóides) para os números naturais.

Exercício 9.22 Desenvolva a prova do Teorema 9.15 - Homomorfismo de Grupóides Preserva a Comutatividade.

Exercício 9.23 Mostre que um homomorfismo de grupóides entre dois monóides não necessariamente preserva o elemento neutro.

Exercício 9.24 Considere o EXEMPLO 9.19 - Homomorfismo: União. Como seria uma prova de que, de fato, o morfismo h sempre preserva a operação?

Exercício 9.25 Verifique se o morfismo abaixo determina um homomorfismo entre as álgebras (e, neste caso, de que tipo de álgebra):

$$f: \langle \mathbf{Z}, *\rangle \to \langle \mathbf{Z}, *\rangle$$

dada por $f(x) = x^2$, sendo $\langle \mathbf{Z}, *\rangle$ o monóide multiplicativo dos inteiros.

Exercício 9.26 Em que condições um monóide livre gerado por um conjunto é um monóide abeliano?

Exercício 9.27 Descreva em palavras o conjunto suporte do monóide livre gerado $\langle \mathbf{N}^*, conc, \varepsilon\rangle$.

Dica: o suporte não é o conjunto gerador **N**.

Exercício 9.28 Considere os monóides livres $\langle \Sigma_1^*, conc_1, \varepsilon\rangle$ e $\langle \Sigma_2^*, conc_2, \varepsilon\rangle$, sendo que $\Sigma_1 = \{a\}$ e $\Sigma_2 = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Então:

a) Apresente um homomorfismo de $\langle \Sigma_2^*, conc_2, \varepsilon\rangle$ para $\langle \Sigma_1^*, conc_1, \varepsilon\rangle$;
b) Defina indutivamente o homomorfismo de monóides canonicamente induzido pela seguinte função entre os conjuntos geradores:

$$f: \Sigma_1 \to \Sigma_2$$

dada por $f(a) = 0$

Exercício 9.29 Sejam $\Sigma_1 = \{a, b\}$ e $\Sigma_2 = \{x, y, z\}$. Considere os monóides livres $\langle \Sigma_1^*, conc_1, \varepsilon\rangle$ e $\langle \Sigma_2^*, conc_2, \varepsilon\rangle$. Então:

a) Defina indutivamente o homomorfismo de monóides canonicamente induzido pela seguinte função entre os conjuntos geradores:

$$f: \Sigma_1 \rightarrow \Sigma_2$$

dada por $f(a) = x$ e $f(b) = y$

b) Apresente um homomorfismo entre os dois monóides o qual *não* é induzido por uma função entre os conjuntos geradores.

Exercício 9.30 Sejam $A = \{a\}$, $B = \{a, b\}$ e $C = \{0, 1, 2\}$. Defina as seguintes relações como grafos, explicitando todas as componentes das correspondentes álgebras:

a) $\varnothing: A \rightarrow A$

b) $\langle B, =\rangle$, dado que = é definida por $\{\langle a, a\rangle, \langle b, b\rangle\}$

c) $\langle C, <\rangle$, dado que < é definida por $\{\langle 0, 1\rangle, \langle 0, 2\rangle, \langle 1, 2\rangle\}$

d) $R: C \rightarrow C$ tal que $R = \{\langle 0, 2\rangle, \langle 2, 0\rangle, \langle 2, 2\rangle\}$

Exercício 9.31 Relativamente aos grafos (o termo "menor" grafo refere-se ao grafo com o menor número possível de nodos e arcos):

a) Qual é o "menor" grafo?

b) Qual o "menor" grafo sendo que o conjunto de nodos é **N**?

Exercício 9.32 Por que nem todo grafo pode ser visto como uma endorrelação?

Dica: é possível definir uma endorrelação para a qual o correspondente grafo possua dois ou mais arcos distintos com o mesmo nodo origem e destino?

Exercício 9.33 Considere os grafos G_1 e G_2 ilustrados na Figura 9.7:

a) Defina um homomorfismo de grafos de G_1 para G_2 diferente do ilustrado;

b) Existe pelo menos um homomorfismo de grafos de G_2 para G_1? Justifique a sua resposta.

Exercício 9.34 Para um grafo G qualquer, qual seria o seu correspondente homomorfismo identidade?

Exercício 9.35 Defina formalmente, detalhando as correspondentes seis-uplas, as "menores" categorias ilustradas na Figura 9.9.

Exercício 9.36 Para determinadas condições, entende-se por "menor" categoria aquela que possui o menor número de objetos e de morfismos tal que satisfaz às condições dadas. Assim, represente diagramaticamente:

a) A "menor" categoria com 3 objetos denotados por A, B e C

b) A "menor" categoria com 3 objetos A, B e C que possua, pelo menos, os morfismos $f: A \rightarrow B$, $g: B \rightarrow C$ e $h: A \rightarrow C$

c) A "menor" categoria com três objetos A, B e C que possua, pelo menos, os morfismos $f: A \rightarrow B$, $g: B \rightarrow C$ e $h: C \rightarrow A$. Compare com a solução do item acima (note-se que a diferença está no morfismo h).

Exercício 9.37 Considere todos os países da América do Sul como objetos e a relação "faz fronteira com" como morfismos. Supondo que um país faz fronteira consigo mesmo, tal construção constitui uma categoria?

Exercício 9.38 Detalhe como um conjunto parcialmente ordenado pode ser visto como uma categoria.

Exercício 9.39 Construa formalmente a Categoria dos Conjuntos das Partes de $\{a, b\}$, denotada por $2^{\{a, b\}}$, na qual:

Objetos: todos os conjuntos de $2^{\{a, b\}}$

Morfismos: todas as funções inclusão sobre os objetos acima.

10 Reticulados e Álgebra Booleana

Este capítulo dá continuidade ao estudo de álgebra, mas em um contexto mais abstrato, motivado pelo estudo realizado no *Capítulo 3 - Álgebra de Conjuntos*, quando foi destacado que a correlação direta existente entre Lógica e Álgebra de Conjuntos não era casual. De fato, ambas são um caso particular de uma álgebra abstrata denominada *Álgebra de Boole* ou *Álgebra Booleana*, originalmente apresentado por George Boole (1815-1864) em 1854.

Uma particularidade das Álgebras Booleanas comparativamente com aquelas apresentadas anteriormente é o fato de que relações de ordem são fundamentais. Com o objetivo de enfatizar a importância das relações de ordem neste contexto, inicialmente é apresentado o conceito de *reticulado* como um conjunto parcialmente ordenado e, somente após, como uma estrutura algébrica. Reticulados são usados para definir Álgebra Booleana.

Reticulados e Álgebras Booleanas são especialmente importantes no contexto da Computação e Informática, bem como na Engenharia e ciência em geral. Em particular, as seguintes aplicações são destacadas:

- *primitivas* para *programação concorrente*;
- *circuitos lógicos* ou *redes lógicas*.

Inicialmente, lembre-se de que uma endorrelação $R: A \rightarrow A$ é uma *Relação de Ordem Parcial Ampla* ou simplesmente *Relação de Ordem Parcial* se R é reflexiva, anti-simétrica e transitiva. Dada uma relação de ordem $\langle A, R\rangle$, as seguintes denominações são usuais:

- A é um *conjunto parcialmente ordenado* ou simplesmente *conjunto ordenado*, eventualmente abreviado por *c.p.o.*;
- A é um *poset* (do inglês, *partially ordered set*).

Lembre-se, também, de que toda relação de ordem pode ser representada na forma de um grafo e, em particular, na forma de um diagrama de Hasse, no qual é usual omitir as arestas que podem ser deduzidas pelas propriedades transitiva e reflexiva (eventualmente, também é comum usar arestas não-orientadas), como ilustrado no seguinte exemplo, anteriormente apresentado.

EXEMPLO 10.1 - Relação como Grafo × Diagrama de Hasse

Considere o conjunto parcialmente ordenado $\langle\{1, 2, 3\}, \leq\rangle$ sendo que a relação $\leq$ é dada pelo seguinte conjunto:

$$\{\langle 1, 1\rangle, \langle 2, 2\rangle, \langle 3, 3\rangle, \langle 1, 2\rangle, \langle 2, 3\rangle, \langle 1, 3\rangle\}$$

Assim, a relação de ordem é representada na Figura 10.1, como grafo (esquerda) e como diagrama de Hasse no qual as arestas podem ser orientadas (centro) ou não-orientadas (direita - observe que os elementos são dirigidos de baixo para cima). ❑

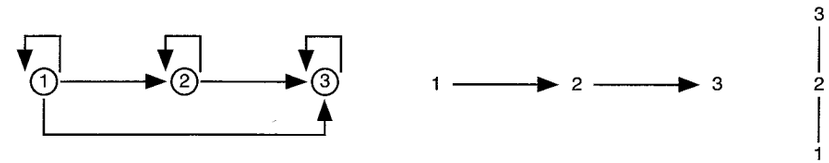


Figura 10.1 Relação: Grafo (esq.) × Diagrama de Hasse orientado (centro) e não-orientado (dir.)

10.1 Limitantes de Conjuntos Parcialmente Ordenados

Antes de apresentar o conceito de reticulado como relação de ordem, é necessário apresentar a noção de limitante de um par de elementos de uma relação de ordem parcial. Resumidamente, para uma dada relação de ordem parcial e para um par de elementos do conjunto em questão, definem-se:

- *limitante inferior*: elemento da relação de ordem que *antecede* os dois elementos do par considerado;
- *limitante superior*: elemento da relação de ordem que *sucede* os dois elementos do par considerado.

Claramente, limitantes inferiores e superiores, se existem, não necessariamente são únicos. Neste contexto, para Computação e Informática, são especialmente importantes o *maior limitante inferior* e o *menor limitante superior*, também denominados de *produto* e de *soma*, respectivamente.

Outro limitante importante no estudo que segue é o de *menor elemento* e o de *maior elemento* de uma relação de ordem parcial, também denominados de *elemento inicial* e de *elemento terminal*, respectivamente. O significado de tais elementos é o sugerido pela própria denominação e pode ser resumido como segue:

- *menor elemento*: elemento da relação de ordem que *antecede* qualquer outro elemento do conjunto;
- *maior elemento*: elemento da relação de ordem que *sucede* qualquer outro elemento do conjunto.

Os conceitos de soma e de produto são definidos para pares de elementos da relação de ordem, induzindo operações binárias. Entretanto, poderiam ser generalizados para seqüências de elementos de qualquer tamanho, inclusive tamanho zero (seqüência vazia), quando os conceitos de soma e de produto coincidem com os de menor e de maior elementos, respectivamente. Ou seja, elementos inicial e terminal correspondem à soma e ao produto de zero elementos, respectivamente. A generalização desses conceitos e a discussão detalhada dessa observação não serão realizadas, pois fogem do escopo deste capítulo.

O leitor atento já deve ter percebido que a soma e o produto são conceitos duais, bem como os elementos inicial e terminal. Tal fato fica claro ao longo desta seção.

Definição 10.1 - Limitante Inferior, Limitante Superior, Produto, Soma
Sejam $\langle P, R\rangle$ uma relação de ordem parcial, $a \in P$ e $b \in P$. Um elemento $p \in P$ é dito:

a) *Maior Limitante Inferior*, *Produto* ou *Ínfimo* de a e b se, simultaneamente:

a.1) p é *limitante inferior* de a e b, ou seja:

$$p\,R\,a \quad \text{e} \quad p\,R\,b$$

a.2) p sucede os demais limitantes inferiores de a e b, ou seja:

$$(\forall q \in P)(q \text{ é limitante inferior de } a \text{ e } b \rightarrow q\,R\,p)$$

b) *Menor Limitante Superior*, *Coproduto*, *Soma* ou *Supremo* de a e b se, simultaneamente:

b.1) p é *limitante superior* de a e b, ou seja:

$$a\,R\,p \quad \text{e} \quad b\,R\,p$$

b.2) p antecede os demais limitantes superiores de a e b, ou seja:

$$(\forall q \in P)(q \text{ é limitante superior de } a \text{ e } b \rightarrow p\,R\,q)$$ ❑

Para uma dada relação de ordem parcial $\langle P, R\rangle$, o produto e a soma do par de elementos a e b são usualmente denotados como segue, respectivamente:

produto: $a \times b$
soma: $a + b$

Como será visto adiante, existe uma estreita relação entre produto e soma com operações da Lógica e da Álgebra de Conjuntos, justificando as seguintes notações, as quais são usuais:

produto: $a \wedge b$ ou $a \cap b$
soma: $a \vee b$ ou $a \cup b$

EXEMPLO 10.2 - Produto e Soma: Cadeia
Considere o conjunto parcialmente ordenado $\langle \{1, 2, 3\}, \leq \rangle$ ilustrado na Figura 10.1. Observe que se trata de uma cadeia (a relação é conexa). Então:

a) Para o par de elementos 1 e 2:

$1 = 1 \times 2$ (produto);
$2 = 1 + 2$ (soma).

Analogamente, para o par de elementos 1 e 3:

$1 = 1 \times 3$ (produto);
$3 = 1 + 3$ (soma).

De fato, em uma cadeia, qualquer par de elementos possui soma e produto (por quê? Neste caso, qual a soma e qual o produto?);

b) Para o par de elementos (repetidos) 2 e 2:

$2 = 2 \times 2$ (produto);
$2 = 2 + 2$ (soma).

De fato, em um conjunto parcialmente ordenado, para qualquer par de elementos repetidos, o elemento repetido será simultaneamente produto e soma (por quê?). ❑

Deve ficar claro que as noções de "inferior" e de "superior" são definidas em relação aos pares *ordenados* da relação de ordem R considerada. Ou seja, tais noções refletem as noções "antecede" e "sucede" em relação a R, respectivamente.

Definição 10.2 - Menor e Maior Elementos, Elementos Inicial e Terminal
Sejam $\langle P, R\rangle$ uma relação de ordem parcial. Um elemento $p \in P$ é dito:

a) *Elemento Inicial* ou *Menor Elemento* da relação de ordem, se:

$$(\forall a \in P)(p\,R\,a)$$

b) *Elemento Terminal* ou *Maior Elemento* da relação de ordem, se:

$$(\forall a \in P)(a\,R\,p)$$ ❑

Em uma relação de ordem parcial, o menor e o maior elementos, se existem, são usualmente denotados pelos dígitos 0 e 1 ou, alternativamente, pelos símbolos $\perp$ e $\top$, respectivamente.

EXEMPLO 10.3 - Inicial e Terminal: Cadeia
Considere o conjunto parcialmente ordenado $\langle \{1, 2, 3\}, \leq \rangle$ ilustrado na Figura 10.1. Então:

1 é elemento inicial (menor elemento);
3 é elemento terminal (maior elemento).

De fato, uma cadeia *finita* e não-vazia sempre possui elementos inicial e terminal (por quê? E se a cadeia for infinita?). ❑

Eventualmente, o elemento inicial pode coincidir com o elemento terminal, como ilustrado no seguinte exemplo. Nesse caso, tal elemento também é denominado de *elemento zero*.

EXEMPLO 10.4 - Elemento Zero

Considere o conjunto parcialmente ordenado $\langle \{a\}, = \rangle$ (como seria o correspondente diagrama de Hasse?) o qual é uma cadeia e, portanto, é um reticulado. Nesse caso:

a é elemento inicial;
a é elemento terminal.

Portanto, a é um elemento zero. ❑

Observação 10.3 - Dualidade: Produto × Soma; Inicial × Terminal

Como já comentado, as noções de produto e de soma são conceitos duais, ou seja, um produto (respectivamente, uma soma), em uma relação de ordem parcial, será uma soma (respectivamente, um produto) na correspondente relação dual.

O mesmo raciocínio também é válido para os elementos inicial e terminal. Como ilustração, dualizando a relação $\langle \{1,2,3\}, \leq \rangle$, ou seja, considerando o conjunto parcialmente ordenado $\langle \{1,2,3\}, \geq \rangle$:

a) Compare com o EXEMPLO 10.2 - Produto e Soma: Cadeia:

$2 = 1 \times 2$ (produto);
$1 = 1 + 2$ (soma);

b) Compare com o EXEMPLO 10.3 - Inicial e Terminal: Cadeia:

1 é elemento terminal;
3 é elemento inicial.

Qual o conceito dual de elemento zero? ❑

Considerando a observação acima e antecipando resultados, os conetivos $\wedge$ e $\vee$, bem como as operações entre conjuntos $\cap$ e $\cup$, também são conceitos duais, assim como os valores-verdade V e F, bem como os conjuntos vazio e universo.

EXEMPLO 10.5 - Soma, Produto, Inicial e Terminal: Conjunto das Partes

Considere o conjunto $A = \{a, b, c\}$ e a relação de ordem $\langle \mathbf{P}(A), \subseteq \rangle$ ilustrada na Figura 10.2. Então:

a) O conjunto $\varnothing$ é elemento inicial, e o conjunto $\{a, b, c\}$ é elemento terminal;
b) Para o par de conjuntos $\{a\}$ e $\{b\}$:

$\varnothing$ é o produto;
$\{a, b\}$ é a soma;

c) Para o par de conjuntos $\{a, b\}$ e $\{a, c\}$:

$\{a\}$ é o produto;
$\{a, b, c\}$ é a soma;

d) Para o par de conjunto $\varnothing$ e $\{a, b, c\}$:

$\varnothing$ é o produto;
$\{a, b, c\}$ é a soma.

Observe que o produto e a soma correspondem às operações de intersecção e de união, respectivamente. Para que o leitor se convença da dualidade das operações de união e de intersecção, sugere-se, como exercício, considerar a relação dual $\langle \mathbf{P}(A), \supseteq \rangle$ e verificar o produto e a soma para os mesmos casos acima. Analogamente para os elementos inicial e terminal. ❑

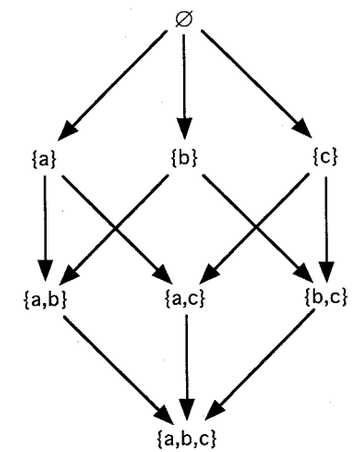


Figura 10.2 Diagrama de Hasse: conjunto das partes

Dependendo da relação de ordem parcial considerada, soma, produto, inicial ou terminal podem ou não existir, conforme ilustrado no exemplo a seguir.

EXEMPLO 10.6 - Soma, Produto, Inicial e Terminal: Concorrência

Considere o conjunto parcialmente ordenado $\langle \text{Ações}, \text{dep} \rangle$ ilustrado na Figura 10.3, sendo que:

$$\text{Ações} = \{c1, c2, c3, p1, p2, q1, q2, q3\}$$

Lembre-se de que essa relação de ordem já foi apresentada anteriormente e representa uma relação de dependência causal entre as ações (elementos do conjunto). Nesse caso, duas ações são ditas *independentes* se não são pares da relação de ordem. Por exemplo:

p1 e q2 são ações independentes;
c1 e p2 são ações dependentes.

A relação de ordem parcial $\langle \text{Ações}, \text{dep} \rangle$ é tal que:

a) Não possui elemento inicial, nem elemento terminal;
b) O par de ações p1 e q1 *não* possui soma nem produto;
c) Para o par de ações c2 e q2:

não possui produto
$c3 = c2 + q2$

d) Para o par de ações p2 e c3:

$c2 = p2 \times c3$
não possui soma

e) Para o par de ações c2 e c3:

$c2 = c2 \times c3$
$c3 = c2 + c3$

Em um conjunto parcialmente ordenado visto como um sistema concorrente, a existência de uma soma ou de um produto possui importantes interpretações, assim como a existência de elemento inicial e terminal (ver 10.5 - Leitura Complementar: Primitivas para Programação Concorrente).❑

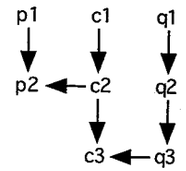


Figura 10.3 Conjunto parcialmente ordenado visto como um sistema concorrente

O resultado do seguinte teorema pode ser intuído dos exemplos apresentados e é especialmente importante no estudo dos reticulados adiante.

Teorema 10.4 - Unicidade: Soma, Produto, Inicial e Terminal
Em um conjunto parcialmente ordenado $\langle P, R\rangle$:

- um elemento inicial (respectivamente, terminal), se existe, é *único*;
- para um dado par de elementos, o produto (respectivamente, a soma), se existe, é *único*.

Prova: (*por absurdo*)
Para provar a unicidade dos elementos inicial e terminal, basta provar para um dos dois elementos, pois a prova para o outro elemento será a dual. Analogamente para soma e produto. A prova que segue é para o elemento inicial (para que o leitor se convença da dualidade, sugere-se, como exercício, detalhar a prova do elemento terminal). A prova da unicidade da soma ou do produto é sugerida como exercício.

Unicidade do elemento inicial. Suponha que 0 e $0'$ são elementos iniciais distintos de $\langle P, R\rangle$. Então:

como 0 é elemento inicial, tem-se que $0\,R\,0'$
analogamente, como $0'$ é elemento inicial, tem-se que $0'\,R\,0$

Pela propriedade anti-simétrica da relação de ordem parcial, tem-se que $0 = 0'$, o que é um absurdo, pois foi suposto que são elementos iniciais distintos.

Logo, o elemento inicial, se existe, é único. ❑

10.2 Reticulados

Reticulados não necessariamente incluem uma ordem na sua estrutura. Entretanto, em um estudo mais aprofundado, é fácil verificar que a definição de reticulado baseada em relação de ordem ou baseada em álgebra são equivalentes. Neste contexto, inicialmente é apresentada a definição de reticulado como relação de ordem e, posteriormente, como uma álgebra. Para mostrar a equivalência entre as duas definições, seria necessário fazer também o caminho inverso: da definição algébrica, concluir a definição baseada em relação de ordem, o que é sugerido como exercício de pesquisa.

10.2.1 Reticulado como Relação de Ordem

Um reticulado é um conjunto parcialmente ordenado no qual todo par de elementos do conjunto possui simultaneamente soma e produto.

Definição 10.5 - Reticulado (definição baseada em c.p.o.)
Seja $\langle P, R\rangle$ uma relação de ordem parcial. Então $\langle P, R\rangle$ é um *Reticulado* se qualquer par de elementos de P possui simultaneamente menor limitante superior (soma) e maior limitante inferior (produto), ou seja:

$$(\forall a \in P)(\forall b \in P)(a \times b \in P \wedge a + b \in P)$$ ❑

Como a soma e o produto são conceitos duais, se o conjunto parcialmente ordenado $\langle P, R\rangle$ é um reticulado, então a relação dual $\langle P, R^{op}\rangle$ também é um reticulado.

EXEMPLO 10.7 - Reticulado: Cadeia
Como, em uma cadeia, qualquer par de elementos possui soma e produto, tem-se que qualquer cadeia é um reticulado (qual a forma geral do diagrama de Hasse de uma cadeia?). Em particular, são reticulados:

$\langle \{1, 2, 3\}, \leq \rangle$
$\langle \mathbf{N}, \geq \rangle$
$\langle \{1\}, = \rangle$

Como fica o caso da relação $\langle \varnothing, \varnothing \rangle$, ou seja, a relação vazia definida sobre o conjunto vazio? É uma cadeia? Nesse caso, é um reticulado? ❑

No exemplo acima, o reticulado $\langle \mathbf{N}, \geq \rangle$ não possui elemento terminal. Conseqüentemente, a existência de inicial ou de terminal não é uma propriedade geral dos reticulados.

EXEMPLO 10.8 - Reticulado × Diagrama de Hasse
Observando o diagrama de Hasse de uma relação de ordem parcial, é possível concluir se a relação é um reticulado. Observe atentamente os itens abaixo e procure justificar cada caso:

a) As relações de ordem ilustradas na Figura 10.4 e na Figura 10.5 são reticulados (possuem elemento inicial e terminal?);
b) As relações de ordem ilustradas na Figura 10.6 *não* são reticulados. Por exemplo, observe que:
   - esquerda: o par de elementos a e b não possui produto;
   - centro: o par de elementos d e e não possui soma;
   - direita: o par de elementos e e g não possui produto (por quê? Possui soma?). ❑

EXEMPLO 10.9 - Reticulado: Conjunto das Partes
Considere o conjunto $A = \{a, b, c\}$ e a relação de ordem parcial $\langle \mathbf{P}(A), \subseteq \rangle$ ilustrada na Figura 10.2. Claramente a relação é um reticulado, pois é um dos casos apresentados na Figura 10.5 (direita).

Portanto, por dualidade, a relação $\langle \mathbf{P}(A), \supseteq \rangle$ também é um reticulado (qual o correspondente diagrama de Hasse?). ❑

EXEMPLO 10.10 - Reticulado: Divisores
Para um número natural positivo n, suponha que D_n denota o conjunto de todos os divisores de n. Por exemplo:

$D_6 = \{1, 2, 3, 6\}$
$D_8 = \{1, 2, 4, 8\}$
$D_{24} = \{1, 2, 3, 4, 8, 12, 24\}$

Cada conjunto de divisores determina uma relação de ordem parcial $\langle D_n, :\rangle$ na qual, para qualquer par de elementos a e b de D_n:

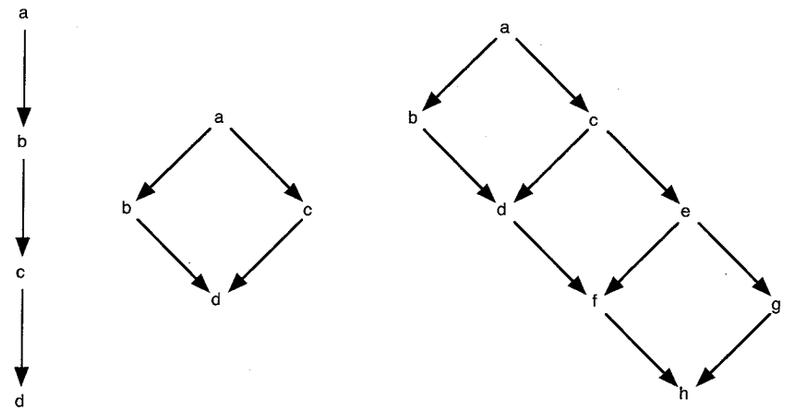


Figura 10.4 Reticulados (diagramas de Hasse)

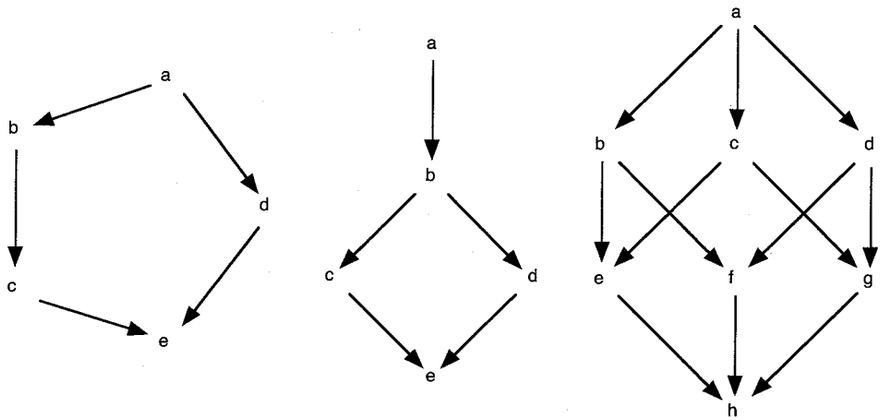


Figura 10.5 Reticulados (diagramas de Hasse)

$$a : b \text{ se e somente se } b \text{ é divisor de } a$$

Por exemplo, para $\langle D_6, : \rangle$, tem-se que 3 é divisor de 6, ou seja, $6 : 3$. Então $\langle D_8, : \rangle$, $\langle D_6, : \rangle$ e $\langle D_{24}, : \rangle$ são reticulados, como ilustrado na Figura 10.7, da esquerda para a direita, respectivamente, pois são casos particulares dos reticulados ilustrados na Figura 10.4. Para um melhor entendimento dos conceitos apresentados, sugere-se como exercício verificar:

- qual o significado da soma e do produto nestes casos?
- qual o correspondente diagrama de Hasse se n for um número primo?
- para qualquer natural positivo n, $\langle D_n, : \rangle$ é um reticulado?
- o mesmo raciocínio vale se o natural considerado for o zero? ❑

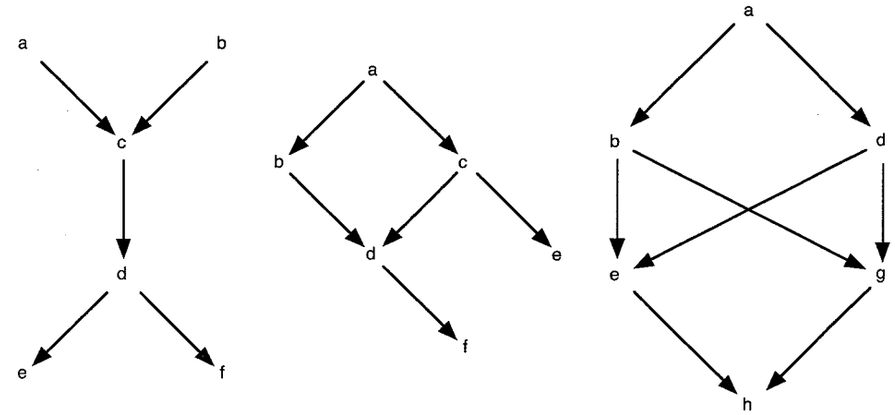


Figura 10.6 Não são reticulados (diagramas de Hasse)

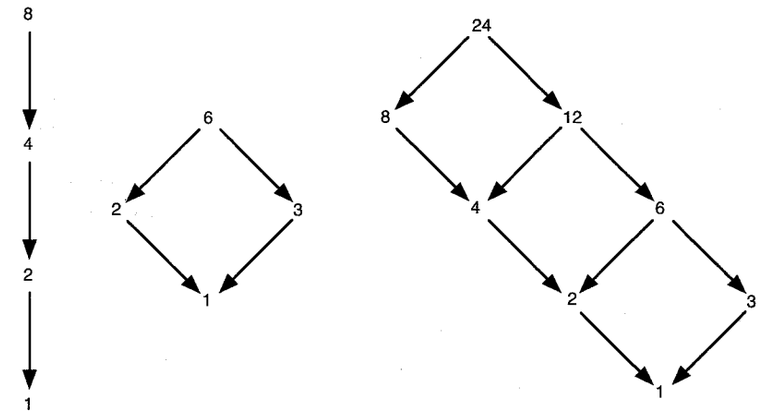


Figura 10.7 Reticulados: divisores de 8 (esquerda), de 6 (centro) e de 24 (direita)

10.2.2 Reticulado como Álgebra

Observe que, em um reticulado $\langle P, R \rangle$, o produto e a soma são conceitos duais e definem duas operações binárias e fechadas $\times : P^2 \to P$ e $+ : P^2 \to P$. Portanto, as seguintes estruturas algébricas constituem grupóides:

$$\langle P, \times \rangle \quad \text{e} \quad \langle P, + \rangle$$

Adicionalmente, para qualquer reticulado $\langle P, R \rangle$, prova-se que $\langle P, \times \rangle$ e $\langle P, + \rangle$:

- satisfazem as propriedades associativa e comutativa. Portanto, constituem semigrupos abelianos;
- satisfazem uma importante propriedade conjunta das duas operações denominada *absorção*, definida como segue (suponha quaisquer a e b elementos de P):

$$a \times (a + b) = a \quad \text{e} \quad a + (a \times b) = a$$

Teorema 10.6 - Reticulado: Propriedade Comutativa
Seja $\langle P, R\rangle$ um reticulado. Então, as operações produto e soma são comutativas.

Prova:
Suponha a e b elementos quaisquer de P. Sejam $p = a \times b$ e $q = b \times a$. Então:

$p = a \times b \Rightarrow$	definição de produto
$p\,R\,a \wedge p\,R\,b \Rightarrow$	comutatividade da conjunção
$p\,R\,b \wedge p\,R\,a \Rightarrow$	definição de produto
$p\,R\,(b \times a) \Rightarrow$	$q = b \times a$
$p\,R\,q$	

Portanto, $p\,R\,q$. Seguindo o mesmo raciocínio para $q = b \times a$, obtem-se que $q\,R\,p$. Como R é anti-simétrica (pois é uma relação de ordem), $p = q$, ou seja:

$$a \times b = b \times a$$

Como $\langle P, R\rangle$ é um reticulado, por dualidade:

$$a + b = b + a$$ ❑

Teorema 10.7 - Reticulado: Propriedade Associativa
Seja $\langle P, R\rangle$ um reticulado. Então, as operações produto e soma são associativas.

Prova:
Suponha a, b e c elementos quaisquer de P. Sejam $p = (a \times b) \times c$ e $q = a \times (b \times c)$. Então:

$p = (a \times b) \times c \Rightarrow$	definição de produto
$p\,R\,(a \times b) \wedge p\,R\,c \Rightarrow$	definição de produto
$(p\,R\,a \wedge p\,R\,b) \wedge p\,R\,c \Rightarrow$	associatividade da conjunção
$p\,R\,a \wedge (p\,R\,b \wedge p\,R\,c) \Rightarrow$	definição de produto
$p\,R\,a \wedge p\,R\,(b \times c) \Rightarrow$	definição de produto
$p\,R\,(a \times (b \times c)) \Rightarrow$	$q = a \times (b \times c)$
$p\,R\,q$	

Portanto, $p\,R\,q$. Seguindo o mesmo raciocínio para $q = a \times (b \times c)$, obtem-se que $q\,R\,p$. Como R é anti-simétrica (pois é uma relação de ordem), $p = q$, ou seja:

$$a \times (b \times c) = (a \times b) \times c$$

Como $\langle P, R\rangle$ é um reticulado, por dualidade:

$$a + (b + c) = (a + b) + c$$ ❑

Teorema 10.8 - Reticulado: Absorção
Seja $\langle P, R\rangle$ um reticulado. Então, as operações produto e soma conjuntamente satisfazem à propriedade absorção.

Prova:
Suponha a, b e c elementos quaisquer de P. Sejam $p = a \times (a + b)$ e $q = a + (a \times b)$. Então:

$p = a \times (a + b) \Rightarrow$	definição de produto
$p\,R\,a \wedge p\,R\,(a + b) \Rightarrow$	simplificação da conjunção
$p\,R\,a$	

Portanto, $p\,R\,a$. Por outro lado, como R é reflexiva, $a\,R\,a$. Então:

$a\,R\,a \Rightarrow$	definição de soma e transitividade de R
$a\,R\,(a + b) \Rightarrow$	definição de produto (a é limitante inferior de a e de $a + b$)
$a\,R\,(a \times (a + b)) \Rightarrow$	$p = a \times (a + b)$
$a\,R\,p$	

Como $p\,R\,a$ e $a\,R\,p$, e considerando que R é anti-simétrica, $a = p$, ou seja:

$$a \times (a + b) = a$$

Como $\langle P, R\rangle$ é um reticulado, por dualidade:

$$a = a + (a \times b)$$ ❑

Definição 10.9 - Reticulado (definição algébrica)
Sejam $\langle P, \times\rangle$ e $\langle P, +\rangle$ dois semigrupos abelianos tais que satisfazem conjuntamente a propriedade absorção. Então a seguinte álgebra monossortida com duas operações binárias é um *Reticulado*:

$$\langle P, \times, +\rangle$$ ❑

EXEMPLO 10.11 - Reticulado (definição algébrica): Conetivos $\wedge$ e $\vee$
Considere os valores-verdade F e V, bem como os conetivos lógicos $\wedge$ e $\vee$. Então a seguinte estrutura constitui um reticulado (qual o correspondente diagrama de Hasse?):

$$\langle \{F, V\}, \wedge, \vee\rangle$$

De fato, quando do estudo da lógica:

- foi verificado que as estruturas $\langle \{F, V\}, \wedge\rangle$ e $\langle \{F, V\}, \vee\rangle$ são semigrupos abelianos (pois são operações fechadas, associativas e comutativas);
- foi discutido que tais conetivos satisfazem à propriedade absorção. ❑

EXEMPLO 10.12 - Reticulado (definição algébrica): Operações $\cap$ e $\cup$
Considere um conjunto A qualquer, bem como as operações $\cap$ e $\cup$. Então a seguinte estrutura constitui um reticulado:

$$\langle \mathbf{P}(A), \cap, \cup\rangle$$

De fato, quando do estudo da álgebra de conjuntos:

- foi verificado que as estruturas $\langle \mathbf{P}(A), \cap\rangle$ e $\langle \mathbf{P}(A), \cup\rangle$ são semigrupos abelianos (pois são operações fechadas, associativas e comutativas);
- foi discutido que tais operações satisfazem a propriedade absorção. ❑

Independentemente das propriedades que originam a definição algébrica de um reticulado, a seguinte propriedade também é satisfeita por qualquer reticulado.

Teorema 10.10 - Reticulado: Idempotência
Seja $\langle P, \times, +\rangle$ um reticulado. Então, as operações produto e soma satisfazem a *propriedade idempotência*, ou seja, para qualquer elemento $p \in P$:

$$p \times p = p \quad \text{e} \quad p + p = p$$

Prova:
Suponha p e q elementos quaisquer de P. Então:

$p \times p = p \Rightarrow$	absorção: $a + (a \times b) = a$, supondo $a = p$ e $b = q$
$p \times (p + (p \times q)) = p \Rightarrow$	absorção: $a \times (a + b) = a$, supondo $a = p$ e $b = p \times q$
$p = p$	

Portanto:

$$p \times p = p \quad \text{e, por dualidade,} \quad p + p = p$$ ❑

EXEMPLO 10.13 - Reticulado: Idempotência
a) Considere o reticulado $\langle \{V, F\}, \wedge, \vee\rangle$. Quando do estudo da lógica, foi verificado que:

$$p \wedge p \Leftrightarrow p \quad \text{e} \quad p \vee p \Leftrightarrow p$$

b) Considere o reticulado $\langle \mathbf{P}(A), \cap, \cup \rangle$. Quando do estudo da álgebra de conjuntos, foi verificado que:

$$X \cap X = X \quad \text{e} \quad X \cup X = X$$ ❑

10.3 Tipos Especiais de Reticulados

Três tipos de reticulados destacam-se, a saber:

- *reticulado distributivo*: a soma se distribui sobre o produto e vice-versa;
- *reticulado limitado*: possui elementos inicial e terminal;
- *reticulado complementado*: reticulado limitado no qual cada elemento possui complemento em relação à soma e ao produto.

10.3.1 Reticulado Distributivo

Definição 10.11 - Reticulado Distributivo
Um reticulado $\langle P, \times, + \rangle$ é dito um *Reticulado Distributivo* se (suponha a, b e c elementos quaisquer de P):

$$a \times (b + c) = (a \times b) + (a \times c) \quad \text{e} \quad a + (b \times c) = (a + b) \times (a + c)$$ ❑

De fato, as duas igualdades que definem um reticulado distributivo são equivalentes, ou seja, é suficiente mostrar *uma* das duas igualdades para provar que o reticulado é distributivo.

EXEMPLO 10.14 - Reticulado Distributivo: Conetivos ∧ e ∨
Considere os valores-verdade F e V. O seguinte reticulado é distributivo:

$$\langle \{F, V\}, \wedge, \vee \rangle$$

De fato, quando do estudo da lógica, foi verificada a distributividade do ∧ sobre o ∨ (e vice-versa). ❑

EXEMPLO 10.15 - Reticulado Distributivo: Operações ∩ e ∪
Considere um conjunto A qualquer. O seguinte reticulado é distributivo:

$$\langle \mathbf{P}(A), \cap, \cup \rangle$$

De fato, quando do estudo da álgebra de conjuntos, foi verificada a distributividade da ∩ sobre o ∪ (e vice-versa). ❑

O seguinte exemplo mostra que nem todo reticulado é distributivo.

EXEMPLO 10.16 - Reticulado Não-Distributivo
Os reticulados ilustrados na Figura 10.8 não são distributivos. De fato, para cada caso, a não-distributividade pode ser verificada por absurdo:

a) Suponha que o reticulado R1 é distributivo. Então, em particular:

$$a + (b \times c) = (a + b) \times (a + c) \Leftrightarrow$$
$$a \times 0 = b + 1 \Leftrightarrow$$
$$0 = 1$$

o que é um absurdo. Logo, R1 é não-distributivo;

b) A prova da não-distributividade do R2 é sugerida como exercício. ❑

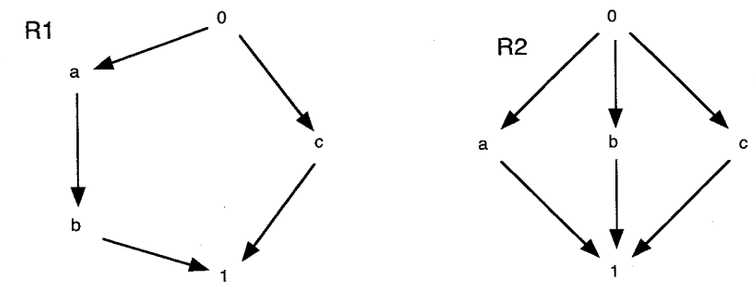


Figura 10.8 Reticulados não-distributivos

No estudo dos homomorfismos de reticulados adiante, é apresentado um teorema geral que auxilia na verificação se um reticulado é ou não distributivo. O seguinte teorema é mais específico e mostra que toda cadeia é um reticulado distributivo.

Teorema 10.12 - Cadeias são Reticulados Distributivos
Toda cadeia é um reticulado distributivo.

Prova:
Sejam $\langle P, \times, + \rangle$ uma cadeia e a, b e c elementos quaisquer de P. Considere os seguintes casos complementares:

Caso 1: a R b *ou* a R c. Então (procure justificar cada caso):

$$a \times (b + c) = a$$
$$(a \times b) + (a \times c) = a$$

Caso 2: b R a *e* c R a. Então (procure justificar cada caso):

$$a \times (b + c) = b + c$$
$$(a \times b) + (a \times c) = b + c$$

Logo, toda cadeia é distributiva. ❑

10.3.2 Reticulado Limitado

Um *reticulado limitado* é um reticulado o qual possui elemento inicial e terminal.

Definição 10.13 - Reticulado Limitado
Um reticulado $\langle P, \times, + \rangle$ é dito um *Reticulado Limitado* se possui elemento inicial 0 e elemento terminal 1. Neste caso, o reticulado é usualmente denotado como segue:

$$\langle P, \times, +, 0, 1 \rangle$$ ❑

EXEMPLO 10.17 - Reticulado Limitado
Os reticulados ilustrados na Figura 10.8 são limitados. ❑

EXEMPLO 10.18 - Reticulado Limitado: Cadeia
Toda cadeia finita e não-vazia é um reticulado limitado. Em particular, a cadeia finita $\langle \{a\}, = \rangle$ é um reticulado na qual a é simultaneamente elemento inicial e terminal (ou seja, é elemento zero). ❑

EXEMPLO 10.19 - Reticulado Não-Limitado
A cadeia $\langle \mathbf{N}, \geq \rangle$ é um reticulado, mas *não* é limitado, pois não tem elemento terminal. ❑

EXEMPLO 10.20 - Reticulado Limitado: Conetivos $\wedge$ e $\vee$
Considere os valores-verdade F e V. O seguinte reticulado é limitado (observe que se trata de uma cadeia finita na qual $F \leq V$):

$$\langle \{F, V\}, \wedge, \vee, F, V \rangle$$ ❑

EXEMPLO 10.21 - Reticulado Limitado: Operações $\cap$ e $\cup$
Considere um conjunto A qualquer. O seguinte reticulado é limitado:

$$\langle \mathbf{P}(A), \cap, \cup, \varnothing, A \rangle$$

Em particular, a relação $\langle \mathbf{P}(\{a, b, c\}), \subseteq \rangle$ ilustrada na Figura 10.2 é um reticulado limitado. ❑

Pela definição de produto e de soma, qualquer reticulado limitado satisfaz as seguintes propriedades:

a) *Elemento Absorvente.*

$$a \times 0 = 0 \quad \text{e} \quad a + 1 = 1 \tag{1}$$

b) *Elemento Neutro.*

$$a \times 1 = a \quad \text{e} \quad a + 0 = a \tag{2}$$

Portanto, as estruturas algébricas $\langle P, \times, 1 \rangle$ e $\langle P, +, 0 \rangle$ constituem monóides comutativos.

10.3.3 Reticulado Complementado

Um *reticulado complementado* é um reticulado limitado no qual cada elemento possui complemento em relação à soma (resultando no elemento terminal) e ao produto (resultando no elemento inicial).

Definição 10.14 - Complemento, Reticulado Complementado
Sejam $\langle P, \times, +, 0, 1 \rangle$ um reticulado limitado e $a \in P$. Então:

a) Um elemento $a' \in P$ é dito *Complemento* de a (e vice-versa) se:

$$a \times a' = 0 \quad \text{e} \quad a + a' = 1$$

b) Um reticulado é dito um *Reticulado Complementado* se qualquer elemento possui um complemento, ou seja:

$$(\forall a \in P)(\exists a' \in P)(a \times a' = 0 \wedge a + a' = 1)$$

Nesse caso, o reticulado é usualmente denotado como segue:

$$\langle P, \times, +, ', 0, 1 \rangle$$ ❑

Considerando as propriedades elemento absorvente (1) e elemento neutro (2) referentes aos reticulados limitados, o elemento terminal é o complemento do inicial (e vice-versa), ou seja:

$$1 \times 0 = 0 \quad \text{e} \quad 1 + 0 = 1$$

Complemento não deve ser confundido com elemento inverso. Conseqüentemente, um reticulado complementado não necessariamente implica que as estruturas algébricas $\langle P, \times, 1 \rangle$ e $\langle P, +, 0 \rangle$ constituam grupos.

EXEMPLO 10.22 - Reticulado Complementado: Lógica
Considere os valores-verdade F e V. O seguinte reticulado é complementado:

$$\langle \{F, V\}, \wedge, \vee, \neg, F, V \rangle$$

De fato, V é o complemento de F, ou seja:

$$F \wedge \neg F = F \wedge V = F \quad \text{e} \quad F \vee \neg F = F \vee V = V$$ ❑

EXEMPLO 10.23 - Reticulado Complementado: Conjuntos
Considere um conjunto A qualquer. A seguinte estrutura algébrica é um reticulado complementado:

$$\langle \mathbf{P}(A), \cap, \cup, \sim, \varnothing, A \rangle$$

Em particular, o conjunto $A = \{a, b, c\}$ e a correspondente relação de ordem $\langle \mathbf{P}(A), \subseteq \rangle$ ilustrada na Figura 10.2 é um reticulado complementado. Por exemplo, o complemento do conjunto $\{a, b\}$ é o conjunto $\{c\}$, ou seja:

$$\{a, b\} \cap \{c\} = \varnothing$$
$$\{a, b\} \cup \{c\} = \{a, b, c\}$$ ❑

O complemento de um elemento, se existe, não necessariamente é único, como ilustrado no seguinte exemplo.

EXEMPLO 10.24 - Complemento Não Necessariamente é Único
Considere o reticulado R2 limitado ilustrado na Figura 10.8 (direita), o qual é complementado. Entretanto, o complemento de um elemento não necessariamente é único. Por exemplo, b e c são *ambos* complementos de a. Sugere-se como exercício verificar se, no reticulado R1, existe algum elemento com mais de um complemento. ❑

10.4 Sub-Reticulado

Intuitivamente, um *sub-reticulado* é uma parte do reticulado a qual também é um reticulado. Sub-reticulados são especialmente importantes, quando do estudo dos homomorfismos de reticulados adiante, pois auxiliam na determinação se determinado reticulado é ou não distributivo.

Definição 10.15 - Sub-Reticulado
Sejam $\langle P, \times, + \rangle$ um reticulado e $Q \subseteq P$. Então $\langle Q, \times, + \rangle$ é um *Sub-Reticulado* de $\langle P, \times, + \rangle$ se $\langle Q, \times, + \rangle$ for um reticulado no qual as operações de produto e de soma são como em $\langle P, \times, + \rangle$, mas restritas a Q. ❑

EXEMPLO 10.25 - Sub-Reticulado

a) Na Figura 10.9, são ilustrados um reticulado (esquerda) e dois correspondentes sub-reticulados (centro e direta);
b) Para o reticulado na Figura 10.9 (esquerda), *não* são correspondentes sub-reticulados os diagramas ilustrados na Figura 10.10 (por quê?). ❑

O pleno entendimento do seguinte exemplo é fundamental para o correto entendimento da definição de sub-reticulado.

EXEMPLO 10.26 - Não-Sub-Reticulado
Considere a Figura 10.11. A relação de ordem parcial $\langle \mathbf{P}(\{a, b, c\}), \subseteq \rangle$ determina o reticulado R1 (esquerda). Observe atentamente que as estruturas R2 (centro) e R3 (direita) constituem reticulados, mas *não* são sub-reticulados de R1 pois:

- para $\{a, b\}$ e $\{b, c\}$, o produto em R1 é $\{b\}$, ao passo que o produto em R2 é $\varnothing$. Portanto, a operação de produto em R2 não é como em R1 para os mesmos elementos;
- para $\{a\}$ e $\{b\}$, a soma em R1 é $\{a, b\}$, ao passo que a soma em R3 é $\{a, b, c\}$. Portanto, a operação de soma em R3 não é como em R1 para os mesmos elementos. ❑

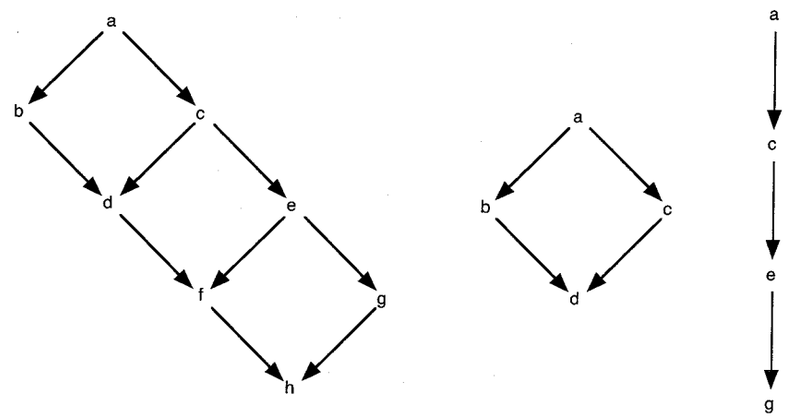


Figura 10.9 Reticulado (esquerda) e sub-reticulados (centro, direita)

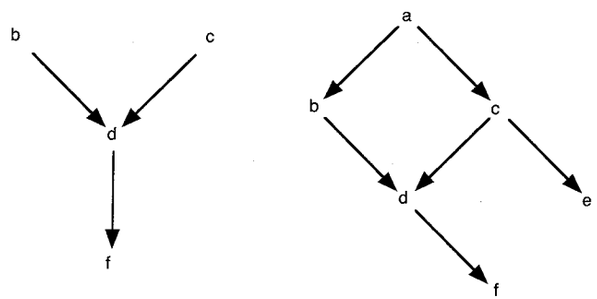


Figura 10.10 Não são sub-reticulados

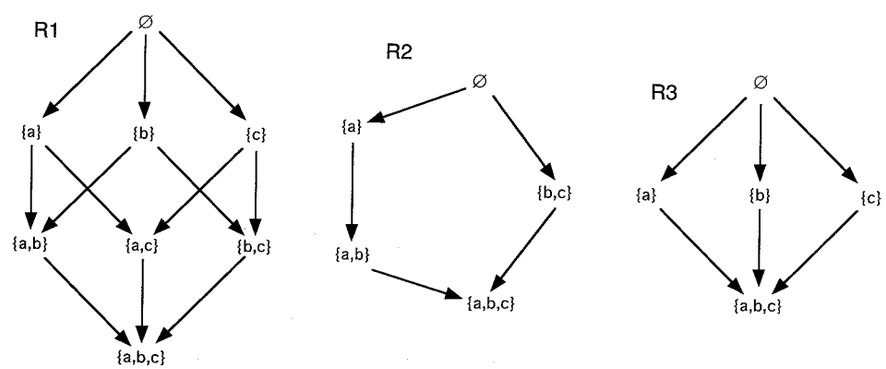


Figura 10.11 Reticulado (esquerda) e reticulados que *não* são sub-reticulados (centro, direita)

10.5 Leitura Complementar: Primitivas para Programação Concorrente

Lembre-se de que conjuntos parcialmente ordenados constituem um modelo semântico para sistemas concorrentes. Nesse contexto, a existência de produto entre dois elementos pode ser interpretada como segue:

a) *Dependência causal.* Os dois elementos são dependentes causais de um mesmo elemento, pois possuem um limitante inferior. Por exemplo, no conjunto parcialmente ordenado $\langle\{a, b, c, p, q, x, y, z\}, \leq\rangle$ ilustrado na Figura 10.12, embora b seja independente de c, ambos dependem de q para ocorrerem. Adicionalmente, como o próprio nome indica, q é o "maior" ("mais recente") elemento dos quais b e c são dependentes causais, ou seja, é o produto de b e c. Note-se que ambos também dependem de p, mas este não é o "maior" (limitante inferior). Adicionalmente, não existe produto entre y e p, significando que não possuem qualquer dependência causal comum;

b) *Concorrência.* Além da dependência causal, b e c são concorrentes. Ou seja, não só dependem da ocorrência de q, como também pertencem a partes independentes do sistema, com origem em q.

Tal interpretação pode ser usada para dar semântica a uma *primitiva* (operação atômica, em geral implementada no *núcleo* de um *sistema operacional*) denominada `fork`, a qual possui como função o "disparo" de partes concorrentes (independentes) de um sistema.

A Figura 10.13 ilustra um trecho de programa em uma pseudolinguagem (tipo *Pascal*) correspondente ao subconjunto parcialmente ordenado ilustrado na Figura 10.12 (esquerda). Nesse caso, os rótulos das instruções correspondem aos elementos do conjunto parcialmente ordenado (o que cada instrução rotulada faz não é detalhado). O trecho de programa ilustra o disparo e o controle de fim de processamento (usando uma outra primitiva denominada `quit`) das partes concorrentes.

Assim, a existência do produto entre dois elementos significa que existe um `fork` que disparou as duas partes independentes do sistema (qual a interpretação se o produto for um dos dois elementos considerados?).

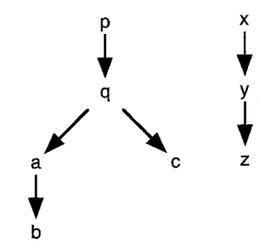


Figura 10.12 Conjunto parcialmente ordenado: dependência causal/concorrência

Analogamente ao produto, a existência da soma entre dois elementos também possui uma importante interpretação, que é a de caracterizar a existência e o momento em que ocorre a sincronização de duas partes independentes (concorrentes) de um sistema (e se não existir a soma?).

```
p:
q:    fork a, c
      quit
a:
b:
      quit
c:
      quit
```

Figura 10.13 Trecho de programa usando as primitivas fork e quit

Tal sincronização pode ser usada para dar semântica a uma primitiva denominada `join` (de fato, em inglês, *join* é uma denominação alternativa de soma). Por exemplo, no conjunto parcialmente ordenado na Figura 10.14, w é a soma de b e u. Assim, as partes independentes do sistema em que se encontram b e u serão sincronizadas em w (qual a interpretação se a soma for um dos dois elementos considerados?), o que significa que existe um `join` em w, como ilustrado no trecho de programa na Figura 10.15 (o número 2 após o `join` significa a sincronização de duas partes independentes). Note-se a dificuldade de expressar um conjunto parcialmente ordenado (como um modelo para sistemas concorrentes) usando as primitivas `fork`, `quit` e `join` apresentadas.

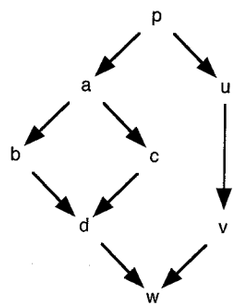


Figura 10.14 Reticulado representando um sistema concorrente

```
p:    fork a, u
      quit
a:    fork b, c
      quit
b:    goto d
c:    goto d
d:    join 2
      goto w
u:
v:
w:    join 2
      quit
```

Figura 10.15 Trecho de programa usando as primitivas fork, quit e join

Como produto e soma são conceitos duais, conclui-se que as primitivas `fork` e `join` também são duais. Por fim, uma importante conclusão desta discussão é que, idealmente, um sistema concorrente deve ser um reticulado. Adicionalmente, freqüentemente é desejado que o reticulado seja complementado, o que é considerado uma boa técnica de construção de sistemas, ou seja, no qual:

- o elemento inicial significa que possui um único início ou ponto de entrada;
- o elemento terminal significa que o sistema tem um único fim, ou seja, existe um ponto de terminação bem definido.

10.6 Álgebra Booleana

Uma Álgebra Booleana é um reticulado especializado particularmente importante para Computação e Informática. Destaque-se que:

- Lógica e Álgebra de Conjuntos são casos particulares da Álgebra Booleana, o que explica a forte correlação entre essas duas álgebras aparentemente tão distintas, conforme é discutido ao final desta seção;
- Álgebra Booleana é usada para modelar circuitos de dispositivos eletrônicos, conforme é comentado adiante.

Definição 10.16 - Álgebra Booleana
Uma *Álgebra Booleana* ou *Álgebra de Boole* é um reticulado distributivo e complementado. ❑

Uma Álgebra Booleana é usualmente denotada como um reticulado complementado:

$$\langle P, \times, +, ', 0, 1 \rangle$$

Pode-se verificar que, considerando que se trata de um reticulado complementado e distributivo, então ele é unicamente complementado. Adicionalmente, pelos estudos anteriores deste capítulo, uma Álgebra Booleana $\langle P, \times, +, ', 0, 1 \rangle$ satisfaz as seguintes propriedades (suponha a, b e c elementos quaisquer de P):

a) Por ser um reticulado:

Associativa.

$$a \times (b \times c) = (a \times b) \times c$$
$$a + (b + c) = (a + b) + c$$

Comutativa.

$$a \times b = b \times a$$
$$a + b = b + a$$

Absorção.

$$a \times (a + b) = a$$
$$a + (a \times b) = a$$

Idempotência.

$$a \times a = a$$
$$a + a = a$$

b) Por ser um reticulado distributivo:

Distributiva.

$$a \times (b + c) = (a \times b) + (a \times c)$$
$$a + (b \times c) = (a + b) \times (a + c)$$

c) Por ser um reticulado complementado (e, conseqüentemente, limitado):

Elemento Neutro.

$$a \times 1 = a$$
$$a + 0 = a$$

Elemento Absorvente.

$$a \times 0 = 0$$
$$a + 1 = 1$$

Complemento.

$$a \times a' = 0$$
$$a + a' = 1$$

d) Adicionalmente, as seguintes propriedades podem ser verificadas (o que é sugerido como exercício):

Duplo Complemento.

$$(a')' = a$$

DeMorgan.

$$(a \times b)' = a' + b'$$
$$(a + b)' = a' \times b'$$

EXEMPLO 10.27 - Álgebra Booleana: Lógica

Considere os valores-verdade F e V. Já foi visto que a seguinte estrutura é um reticulado distributivo e complementado, ou seja, é uma Álgebra Booleana:

$$\langle \{F, V\}, \wedge, \vee, \neg, F, V \rangle$$ ❑

EXEMPLO 10.28 - Reticulado Limitado: Conjuntos

Considere um conjunto A qualquer. Já foi visto que a seguinte estrutura é um reticulado distributivo e complementado, ou seja, é uma Álgebra Booleana:

$$\langle \mathbf{P}(A), \cap, \cup, \varnothing, A \rangle$$

Em particular, a relação $\langle \mathbf{P}(\{a, b, c\}), \subseteq \rangle$ ilustrada na Figura 10.2 é uma Álgebra Booleana. ❑

Observação 10.17 - Lógica × Álgebra de Conjuntos

Como antecipado no Capítulo 3 - Álgebra de Conjuntos, bem como em diversos pontos ao longo deste livro, a correlação direta entre *Lógica Matemática* e *Álgebra de Conjuntos* não é casual. De fato, como pode ser observado nos dois exemplos acima, as duas álgebras são casos particulares da Álgebra Booleana. Conseqüentemente, ambas herdam todas as propriedades desse tipo de álgebra. ❑

Portanto, uma das vantagens em verificar se uma determinada estrutura algébrica constitui uma Álgebra Booleana é o fato de que todas as propriedades listadas acima (bem como outras não estudadas neste livro) são automaticamente decorrentes.

Entretanto, deve ficar claro para o leitor que nem todas as propriedades listadas são independentes. Ou seja, algumas podem ser obtidas a partir de outras, como, por exemplo, as

propriedades duplo complemento e DeMorgan. Portanto, definições alternativas de álgebra Booleana são freqüentemente apresentadas, usando um conjunto de propriedades independentes. Como ilustração, é apresentada, a seguir, *uma* definição alternativa (entre diversas outras usuais).

Definição 10.18 - Álgebra Booleana (definição alternativa)

Uma *Álgebra Booleana* ou *Álgebra de Boole* é uma álgebra monossortida:

$$\langle P, \times, +, ', 0, 1 \rangle$$

na qual:

Produto. Operação binária interna e fechada tal que $\times : P^2 \rightarrow P$;
Soma. Operação binária interna e fechada tal que $+ : P^2 \rightarrow P$;
Complemento. Operação unária interna e fechada tal que $' : P \rightarrow P$;
Elemento Inicial. Elemento distinguido $0 \in P$;
Elemento Terminal. Elemento distinguido $1 \in P$;

tal que satisfaz às seguintes propriedades (suponha a, b e c elementos quaisquer de P):

Comutativa.

$$a \times b = b \times a$$
$$a + b = b + a$$

Distributiva.

$$a \times (b + c) = (a \times b) + (a \times c)$$
$$a + (b \times c) = (a + b) \times (a + c)$$

Elemento Neutro.

$$a \times 1 = a$$
$$a + 0 = a$$

Complemento.

$$a \times a' = 0$$
$$a + a' = 1$$ ❑

10.7 Circuitos Lógicos

O matemático americano Claude Shannon identificou, em 1938, uma forte correlação entre a lógica proposicional e a lógica de circuitos e foi o primeiro a sugerir que a Álgebra Booleana poderia unificar as duas abordagens, servindo como formalismo para modelar *circuitos lógicos* ou *redes lógicas*. Uma importante conseqüência desse trabalho é que os circuitos lógicos podem ser modelados, analisados, testados e otimizados independentemente de sua implementação.

De fato, praticamente todos os circuitos de um computador digital são modelados em termos de uma Álgebra Booleana cujo conjunto suporte possui dois elementos: verdadeiro e falso.

É claro que o projeto de computadores digitais deve considerar diversos outros fatores como o tempo de propagação de um sinal, minimização de problemas decorrentes de falhas de alguns componentes, etc. O texto que segue considera somente os aspectos referentes à Álgebra Booleana, os quais são apenas brevemente apresentados, bem como se concentra em dois tipos de portas e um inversor, a saber (ver Figura 10.16):

- *porta E*, correspondendo ao produto (conjunção), usualmente denotada pelo símbolo "•";
- *porta OU*, correspondendo à soma (disjunção), usualmente denotada pelo símbolo "+";
- *inversor*, eventualmente denominado de *porta NÃO*, correspondendo ao complemento (negação), usualmente denotado pelo símbolo "′".

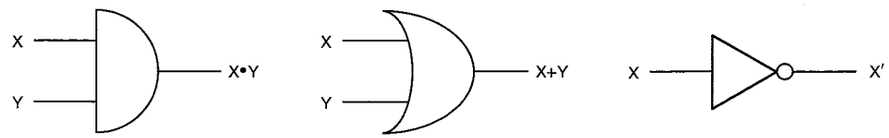


Figura 10.16 Portas: E (esquerda), OU (centro) e NÃO (direita)

Portanto, trata-se da seguinte Álgebra Booleana, na qual 0 e 1 denotam falso e verdadeiro, respectivamente:

$$\langle\{0,1\},\bullet,+,',0,1\rangle$$

O projeto de circuitos é introduzido via exemplos, sendo que:

- sinais (representados por linhas) podem ser combinadas exclusivamente nas portas;
- um sinal pode ser entrada para mais de uma porta. Nesse caso, um ponto de "divisão" é representado por um pequeno círculo preto;
- a saída de uma porta não pode ser entrada para ela mesma (não existem endoarcos).

EXEMPLO 10.29 - Circuito Lógico

A Figura 10.17 representa o diagrama correspondente à seguinte expressão:

$$(X \bullet Y) + Z'$$ ❑

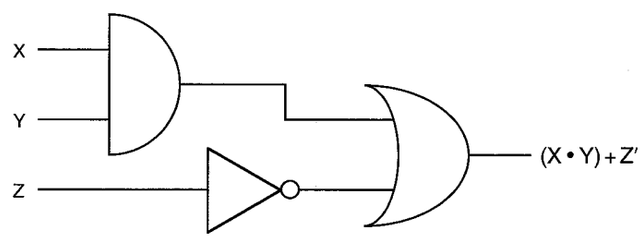


Figura 10.17 Circuito lógico

EXEMPLO 10.30 - Circuito com Múltiplas Saídas: half-adder

A Figura 10.18 representa o diagrama para o *half-adder*, o qual adiciona dois bits X e Y, gerando *duas* saídas, como segue:

- um bit correspondente à (meia) soma dada por $(X+Y)\bullet(X\bullet Y)'$, ou seja:
  0, se $(X=0 \wedge y=0) \vee (X=1 \wedge Y=1)$
  1, caso contrário
- um bit correspondente ao *carry* ("vai um") dado por $X \bullet Y$, ou seja:
  1, se $X=1 \wedge Y=1$
  0, caso contrário ❑

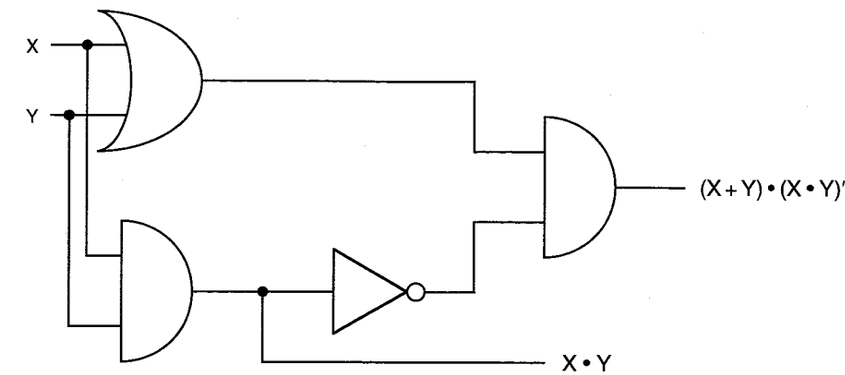


Figura 10.18 Circuito com múltiplas saídas: *half-adder*

Claramente, expressões diferentes (e com diferentes diagramas de circuitos) podem ser equivalentes, ou seja, podem implementar a mesma função lógica. Assim, outro ponto de fundamental importância na construção de circuitos, são as técnicas de minimização de expressões de acordo com determinada forma de expressão. Entretanto, a discussão deste tópico foge do escopo deste livro.

Por fim, portas lógicas são formadas basicamente por transistores para os quais existem diferentes tecnologias de fabricação. Um importante exemplo é a tecnologia *MOS* (Metal Óxido Semicondutor), a qual teve um grande desenvolvimento visando os circuitos em alta escala de integração. Como curiosidade, destaque-se que as portas lógicas típicas desta tecnologia são as negadas como, por exemplo, a *porta NAND*. Assim, para construir uma porta E, é necessário adicionar um inversor na saída da porta NAND.

10.8 Homomorfismos

Quando da apresentação do conceito de álgebra, foi destacado que tão importante quanto o estudo das álgebras é o estudo dos homomorfismos entre álgebras. De fato, o estudo dos homomorfismos não é restrito apenas a estruturas algébricas, mas se estende a qualquer tipo de estrutura, inclusive às não-baseadas em conjuntos, como exemplificado quando da apresentação do conceito de categoria. Em particular, no texto que segue, são apresentados os seguintes homomorfismos:

- *Homomorfismo de Conjuntos Parcialmente Ordenados*, usualmente denominado de *Função Monotônica*;
- *Homomorfismo de Reticulados*;
- *Homomorfismo de Álgebras Booleanas*.

Analogamente ao estudo de homomorfismos realizado no Capítulo 9 - Álgebras e Homomorfismos, a correta extensão dos conceitos de *monomorfismo* e de *epimorfismo* exige noções e conceitos baseados em *Teoria das Categorias* os quais fogem do escopo deste livro. Entretanto, o conceito de *isomorfismo*, baseado na existência de um morfismo inverso, pode ser aplicado a todos os tipos de morfismos estudados a seguir. Neste contexto, duas estruturas são ditas *isomorfas* se existe um isomorfismo entre tais estruturas. Neste caso, as estruturas são

consideradas basicamente a mesma, ou seja, *iguais a menos de isomorfismo* e, obrigatoriamente, possuem as mesmas propriedades.

10.8.1 Homomorfismo de C.P.O. ou Função Monotônica

Um *homomorfismo de conjuntos parcialmente ordenados* ou uma *função monotônica* é uma função que mapeia os conjuntos, preservando a estrutura de ordem. Conjuntos parcialmente ordenados, juntamente com os correspondentes homomorfismos, constituem uma categoria com importantes aplicações no estudo dos sistemas concorrentes, circuitos lógicos, etc.

Definição 10.19 - Homomorfismo de C.P.O., Função Monotônica
Sejam $\langle A, R\rangle$ e $\langle B, S\rangle$ dois conjuntos parcialmente ordenados. Um *Homomorfismo de Conjuntos Parcialmente Ordenados* ou uma *Função Monotônica*, denotado como segue:

$$h: \langle A, R\rangle \rightarrow \langle B, S\rangle$$

é uma função $h: A \rightarrow B$ tal que:

$$(\forall a_1 \in A)(\forall a_2 \in A)(\, a_1 \, R \, a_2 \rightarrow h\langle a_1\rangle \, S \, h\langle a_2\rangle\,) \qquad \square$$

Portanto, uma função monotônica é uma função que mapeia os conjuntos, preservando a estrutura de ordem, no seguinte sentido (veja a Figura 10.19):

se dois elementos estão ordenados no conjunto origem,
então as imagens dos dois elementos estão ordenadas no conjunto destino.

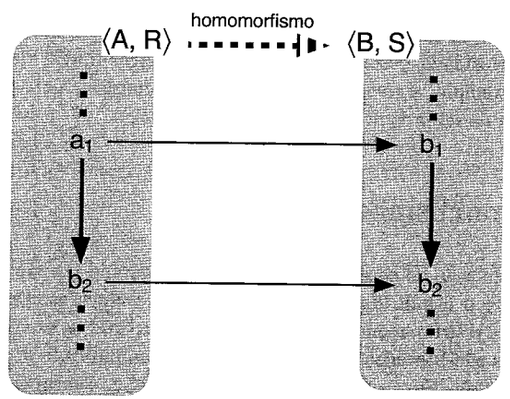


Figura 10.19 Homomorfismo de conjuntos parcialmente ordenados

O teorema a seguir mostra que a composição de funções monotônicas resulta em uma função monotônica. Portanto, conjuntos parcialmente ordenados como objetos e funções monotônicas como morfismos constituem uma categoria.

Teorema 10.20 - Composição de Funções Monotônicas
Sejam $\langle A, R\rangle$, $\langle B, S\rangle$ e $\langle C, T\rangle$ conjuntos parcialmente ordenados e $f: \langle A, R\rangle \rightarrow \langle B, S\rangle$ e $g: \langle B, S\rangle \rightarrow \langle C, T\rangle$ funções monotônicas. Então, o morfismo composto:

$$g \circ f: \langle A, R\rangle \rightarrow \langle C, T\rangle$$

induzido pela composição das funções $f: A \rightarrow B$ e $g: B \rightarrow C$, ou seja, pela função composta $g \circ f: A \rightarrow C$, é uma função monotônica.

Prova:
Suponha que $f: \langle A, R\rangle \rightarrow \langle B, S\rangle$ e $g: \langle B, S\rangle \rightarrow \langle C, T\rangle$ são funções monotônicas. Como a composição de funções é uma função, para mostrar que $g \circ f: \langle A, R\rangle \rightarrow \langle C, T\rangle$ é uma função monotônica, basta mostrar que a composição preserva a ordem. Assim, para quaisquer $a_1 \in A$ e $a_2 \in A$, vale:

$a_1 \, R \, a_2 \Rightarrow$	f é função monotônica
$f\langle a_1\rangle \, S \, f\langle a_2\rangle \Rightarrow$	g é função monotônica
$g\langle f\langle a_1\rangle\rangle \, T \, g\langle f\langle a_2\rangle\rangle \Rightarrow$	definição de composição
$g \circ f\langle a_1\rangle \, T \, g \circ f\langle a_2\rangle$	

Portanto, a ordem é preservada pela função composta. Logo, $g \circ f: \langle A, R\rangle \rightarrow \langle C, T\rangle$ é uma função monotônica. ❑

Observação 10.21 - Categoria Poset
A categoria **Poset** é tal que:

- objetos: todos os conjuntos parcialmente ordenados;
- morfismos: todas as funções monotônicas;
- composição: composição de funções;
- identidade: dada pelas funções monotônicas identidade. ❑

10.8.2 Homomorfismo de Reticulados

A definição de homomorfismo de reticulados que segue é baseada na definição algébrica. Posteriormente, é verificado que a ordem dos correspondentes conjuntos parcialmente ordenados é preservada (o que seria um resultado esperado). Reticulados, juntamente com os correspondentes homomorfismos, constituem uma categoria.

Definição 10.22 - Homomorfismo de Reticulados
Sejam $\langle P, \times, +\rangle$ e $\langle Q, \wedge, \vee\rangle$ dois reticulados. Um *Homomorfismo de Reticulados*, denotado como segue:

$$h: \langle P, \times, +\rangle \rightarrow \langle Q, \wedge, \vee\rangle$$

é uma função entre os conjuntos suportes $h: P \rightarrow Q$ tal que são homomorfismos de semigrupos:

$$h: \langle P, \times\rangle \rightarrow \langle Q, \wedge\rangle \quad \text{e} \quad h: \langle P, +\rangle \rightarrow \langle Q, \vee\rangle \qquad \square$$

Portanto, um homomorfismo de reticulados $h: \langle P, \times, +\rangle \rightarrow \langle Q, \wedge, \vee\rangle$ é dado por uma função entre os conjuntos suportes $h: P \rightarrow Q$ tal que preserva *simultaneamente* as duas operações (produto e soma). Claramente, se apenas uma das operações for preservada, não constitui homomorfismo de reticulados.

Para verificar que um homomorfismo de reticulados $h: \langle P, \times, +\rangle \rightarrow \langle Q, \wedge, \vee\rangle$ preserva a ordem, suponha que $\langle P, R\rangle$ e $\langle Q, S\rangle$ são os correspondentes reticulados, usando a definição baseada em c.p.o. Então, para quaisquer $a \in P$ e $b \in P$, vale:

$a \, R \, b \Leftrightarrow$	operação de produto
$a \times b = a \Rightarrow$	
$h\langle a \times b\rangle = h\langle a\rangle \Rightarrow$	h preserva produto

$h\langle a\rangle \wedge h\langle b\rangle = h\langle a\rangle \Leftrightarrow$ operação de produto
$h\langle a\rangle S h\langle b\rangle$

Portanto, $h: \langle P, R\rangle \rightarrow \langle Q, S\rangle$ preserva a ordem, ou seja:

$$a R b \Rightarrow h\langle a\rangle S h\langle b\rangle$$

Lembre-se de que dois reticulados são ditos *reticulados isomorfos* se existe um isomorfismo de reticulados entre eles e, nesse caso, são basicamente o mesmo reticulado, ou seja, são iguais a menos de isomorfismo. Isso significa que reticulados isomorfos possuem a mesma "forma" (geralmente representada usando diagrama de Hasse). Como ilustração desse fato, reveja os seguintes exemplos, os quais usaram intuitivamente o conceito de isomorfismo de reticulados:

- EXEMPLO 10.8 - Reticulado × Diagrama de Hasse;
- EXEMPLO 10.9 - Reticulado: Conjunto das Partes;
- EXEMPLO 10.10 - Reticulado: Divisores.

O teorema que segue também é baseado em isomorfismos de reticulados, ou seja, na análise da "forma". O teorema não será demonstrado.

Teorema 10.23 - Reticulado Distributivo/Não-Distributivo
Um reticulado não é distributivo se e somente se existir um sub-reticulado isomorfo a um dos dois reticulados ilustrados na Figura 10.8. ❑

Relativamente à composição de homomorfismos de reticulados, considere que:

- como um homomorfismo de reticulados é dado por uma função entre os conjuntos suportes tal que preserva as operações de produto e de soma, ou seja, são dois homomorfismos de semigrupos;
- como a composição de homomorfismos de semigrupos é um homomorfismo de semigrupos;

então a composição de homomorfismos de reticulados é um homomorfismo de reticulados.

Observação 10.24 - Categoria Lattice
A categoria **Lattice** é tal que:

- objetos: todos os reticulados;
- morfismos: todos os homomorfismos de reticulados;
- composição: composição de homomorfismos de reticulados;
- identidade: dada pelos homomorfismos de reticulados identidade. ❑

10.8.3 Homomorfismo de Álgebras Booleanas

Um homomorfismo de álgebras Booleanas, usualmente denominado de homomorfismo Booleano:

$$h: \langle P_1, \times_1, +_1, ', 0_1, 1_1\rangle \rightarrow \langle P_2, \times_2, +_2, \sim, 0_2, 1_2\rangle$$

intuitivamente deve preservar a estrutura das álgebras, ou seja, é tal que, para quaisquer $a \in P_1$ e $b \in P_1$:

$h\langle a \times_1 b\rangle = h\langle a\rangle \times_2 h\langle b\rangle$
$h\langle a +_1 b\rangle = h\langle a\rangle +_2 h\langle b\rangle$
$h\langle a'\rangle = (h\langle a\rangle)\sim$
$h\langle 0_1\rangle = 0_2$
$h\langle 1_1\rangle = 1_2$

Entretanto, tal definição pode ser simplificada pela preservação de apenas duas operações, como segue.

Definição 10.25 - Homomorfismo Booleano
Sejam $\langle P_1, \times_1, +_1, ', 0_1, 1_1\rangle$ e $\langle P_2, \times_2, +_2, \sim, 0_2, 1_2\rangle$ duas álgebras Booleanas. Um *Homomorfismo de Álgebras Booleanas* ou *Homomorfismo Booleano*, denotado como segue:

$$h: \langle P_1, \times_1, +_1, ', 0_1, 1_1\rangle \rightarrow \langle P_2, \times_2, +_2, \sim, 0_2, 1_2\rangle$$

é uma função entre os conjuntos suportes $h: P_1 \rightarrow P_2$ tal que, *alternativamente*:

a) Preserva as operações de produto e de complemento;
b) Preserva as operações de soma e de complemento. ❑

A verificação de que a preservação das operações de complemento e de produto (respectivamente, de soma) implica a preservação da soma (respectivamente, do produto), do elemento inicial e do elemento terminal não é difícil e é sugerida como exercício. Da mesma forma, é fácil verificar que a composição de homomorfismos Booleanos é um homomorfismo Booleano, o que induz a seguinte categoria.

Observação 10.26 - Categoria Bool
A categoria **Bool** é tal que:

- objetos: todas as álgebras Booleanas;
- morfismos: todos os homomorfismos Booleanos;
- composição: composição de homomorfismos Booleanos;
- identidade: dada pelos homomorfismos Booleanos identidade. ❑

10.9 Exercícios

Exercício 10.1 Suponha a, b e c elementos distintos de um dado conjunto A. Mostre que, relativamente a diagramas de Hasse:

a) A configuração ilustrada na Figura 10.20 (esquerda) não pode aparecer em qualquer relação de ordem;
b) A configuração ilustrada na Figura 10.20 (direita) não pode aparecer em qualquer relação de ordem.

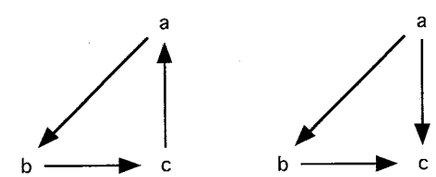


Figura 10.20 Diagramas de Hasse

Exercício 10.2 Considere a Figura 10.4 e a Figura 10.5. Para cada diagrama:

a) Qual o elemento inicial?
b) Qual o elemento terminal?

Exercício 10.3 Considere a Figura 10.6. Para cada diagrama:

a) Verifique a existência de elemento inicial. Justifique a sua resposta;
b) Verifique a existência de elemento terminal. Justifique a sua resposta.

Exercício 10.4 Em uma cadeia não-vazia:

a) Qualquer par de elementos possui soma? Nesse caso, qual é a soma?
b) Qualquer par de elementos possui produto? Nesse caso, qual é o produto?

Exercício 10.5 Justifique:

em um conjunto parcialmente ordenado, para qualquer par de elementos repetidos, o elemento repetido será simultaneamente produto e soma

Exercício 10.6 Relativamente às cadeias:

a) Qual a forma geral do diagrama de Hasse de uma cadeia?
b) $\langle \varnothing, \varnothing \rangle$ é uma cadeia? Nesse caso, possui elemento inicial ou terminal?
c) Se a cadeia for finita e não-vazia:
   - qual o elemento inicial?
   - qual o elemento terminal?
d) Se a cadeia for infinita, obrigatoriamente existe elemento inicial ou elemento terminal?
e) Em que condições uma cadeia possui elemento zero?

Exercício 10.7 Qual o conceito dual de elemento zero?

Exercício 10.8 Considere o conjunto $A = \{a, b, c\}$ e a relação de ordem $\langle \mathbf{P}(A), \supseteq \rangle$. Então (compare com o EXEMPLO 10.5 - Soma, Produto, Inicial e Terminal: Conjunto das Partes):

a) Quais são os elementos inicial e terminal?
b) Para cada par de conjuntos que segue, determine a soma e o produto:
   b.1) $\{a\}$ e $\{b\}$
   b.2) $\{a, b\}$ e $\{a, c\}$
   b.3) $\varnothing$ e $\{a, b, c\}$

Exercício 10.9 Considere o conjunto $A = \{a, b, c\}$ e a relação de ordem $\langle \mathbf{P}(A), R \rangle$ ilustrada na Figura 10.21 na forma de diagrama de Hasse. Então:

a) Verifique a existência dos elementos inicial e terminal. Justifique a sua resposta;
   Dica: observe que o elemento $\{a, b, c\}$ não é origem, nem destino de qualquer aresta do diagrama;
b) Para cada par de conjuntos que segue, determine a soma e o produto:
   b.1) $\{a\}$ e $\{b\}$
   b.2) $\{a, b\}$ e $\{a, c\}$
   b.3) $\varnothing$ e $\{a, b, c\}$
   b.4) $\{a, b, c\}$ e $\{a, b, c\}$

Exercício 10.10 Considere a Figura 10.6. Para cada diagrama:

a) Identifique quais pares de elementos *não* possuem soma;
b) Identifique quais pares de elementos *não* possuem produto.

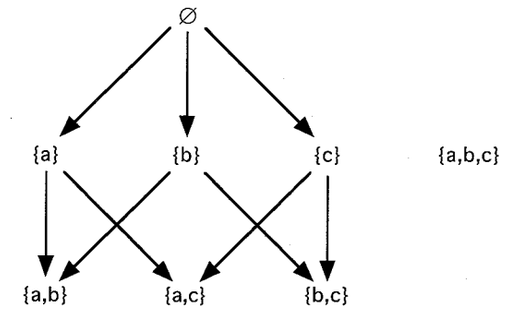


Figura 10.21 Diagrama de Hasse: conjunto das partes

Exercício 10.11 Relativamente ao Teorema 10.4 - Unicidade: Soma, Produto, Inicial e Terminal, detalhe as seguintes provas:

a) Unicidade do elemento terminal.
   Dica: dualize a prova do elemento inicial;
b) Unicidade da soma ou do produto (apenas uma, pois a outra é a prova dual).

Exercício 10.12 Foi afirmado que as duas definições de reticulado apresentadas (como uma relação de ordem e como álgebra) são equivalentes. Da definição baseada em relação de ordem, foi inferida a definição algébrica. Sugere-se, como exercício de pesquisa, a construção do caminho inverso: da definição algébrica, concluir a definição baseada em relação de ordem.

Exercício 10.13 A relação $\langle \varnothing, \varnothing \rangle$ é um reticulado?

Exercício 10.14 Para a relação $\langle D_n, : \rangle$ (do EXEMPLO 10.10 - Reticulado: Divisores):

a) Qual o significado da soma e do produto?
b) Qual o correspondente diagrama de Hasse se n for um número primo?
c) Para qualquer natural positivo n, $\langle D_n, : \rangle$ é um reticulado?
d) $\langle D_0, : \rangle$ é um reticulado?

Exercício 10.15 Relativamente ao seguinte reticulado algebricamente definido:

$$\langle \{F, V\}, \wedge, \vee \rangle$$

a) Qual a correspondente relação de ordem parcial?
b) Faça o diagrama de Hasse da relação de ordem parcial associada.

Exercício 10.16 Prove que o reticulado R2 ilustrado na Figura 10.8 não é distributivo.

Exercício 10.17 Suponha $\langle A, R \rangle$ e $\langle B, S \rangle$ duas cadeias quaisquer. Relativamente às cadeias e aos reticulados limitados:

a) Em quais casos uma cadeia *não* é um reticulado limitado?
b) A união de cadeias $\langle A \cup B, R \cup S \rangle$ pode ser um reticulado limitado? Se pode, exemplifique;
c) A intersecção de cadeias $\langle A \cap B, R \cap S \rangle$ é sempre um reticulado limitado? Justifique.

Exercício 10.18 Para qualquer conjunto A:

a) $\langle \mathbf{P}(A), \supseteq \rangle$ é um reticulado limitado?
b) A união das relações $\langle \mathbf{P}(A), \supseteq \rangle$ e $\langle \mathbf{P}(A), \subseteq \rangle$, ou seja, $\langle \mathbf{P}(A), \supseteq \cup \subseteq \rangle$, é um reticulado limitado?

c) A intersecção das relações $\langle \mathbf{P}(A), \supseteq \rangle$ e $\langle \mathbf{P}(A), \subseteq \rangle$, ou seja, $\langle \mathbf{P}(A), \supseteq \cap \subseteq \rangle$, é um reticulado limitado?

Exercício 10.19 Uma cadeia pode ser um reticulado complementado? Nesse caso, em que condições?

Exercício 10.20 Para a relação $\langle D_{36}, : \rangle$ (do EXEMPLO 10.10 - Reticulado: Divisores):

a) Qual o correspondente diagrama de Hasse?
b) O reticulado é complementado? Justifique a sua resposta.

Exercício 10.21 Para uma dada relação de ordem, entende-se por um *intervalo* nessa relação todos os elementos compreendidos entre os limites fornecidos, respeitando a relação de ordem. Adicionalmente, o intervalo é dito um *intervalo fechado* (respectivamente, *intervalo aberto*), se o limite considerado for elemento (respectivamente, não for elemento) do intervalo em questão. Assim, por exemplo, considerando a relação de ordem $\langle \mathbf{R}, \leq \rangle$:

- $A = [1, 2]$ denota o intervalo *fechado* $A = \{x \in \mathbf{R} \mid x \geq 1 \text{ e } x \leq 2\}$
- $B = (1, 2)$ denota o intervalo *aberto* $B = \{x \in \mathbf{R} \mid x > 1 \text{ e } x < 2\}$
- $C = (1, 2]$ denota o seguinte intervalo *aberto à esquerda* e *fechado à direita*:

$$C = \{x \in \mathbf{R} \mid x > 1 \text{ e } x \leq 2\}$$

Nesse contexto, considerando que os intervalos A, B e C são reticulados, discuta se são:

a) Cadeias;
b) Finitos;
c) Limitados;
d) Complementados.

Exercício 10.22 Relativamente ao cardinal do conjunto suporte de um reticulado complementado:

a) Qual o menor reticulado complementado em termos do cardinal do conjunto suporte?
b) O cardinal do conjunto suporte de um reticulado complementado pode ser infinito?

Exercício 10.23 Verifique se, no reticulado R1 ilustrado na Figura 10.8, existe algum elemento com mais de um complemento.

Exercício 10.24 Para o reticulado ilustrado na Figura 10.9 (esquerda), justifique por que os diagramas ilustrados na Figura 10.10 não são sub-reticulados.

Exercício 10.25 Determine todos os sub-reticulados dos seguintes reticulados:

a) Reticulado ilustrado na Figura 10.4 (esquerda);
b) Reticulado ilustrado na Figura 10.11 (esquerda).

Exercício 10.26 Considerando conjuntos parcialmente ordenados como um modelo semântico para sistemas concorrentes:

a) A existência do produto entre dois elementos do c.p.o. significa que existe um `fork` que disparou as duas partes independentes do sistema. Qual a interpretação se o produto for um dos dois elementos considerados?
b) Qual a interpretação da inexistência da soma entre dois elementos do c.p.o.?
c) A semântica da primitiva `join` é dada pela soma. Qual a interpretação se a soma for um dos dois elementos considerados?

Exercício 10.27 Verifique as seguintes propriedades para uma Álgebra Booleana $\langle P, \times, +, ', 0, 1 \rangle$ qualquer (suponha a, b e c elementos quaisquer de P):

a) *Duplo Complemento.*

$$(a')' = a$$

b) *DeMorgan.*

$$(a \times b)' = a' + b'$$
$$(a + b)' = a' \times b'$$

Exercício 10.28 Mostre que, para quaisquer elementos a, b e c de P, e para qualquer Álgebra Booleana $\langle P, \times, +, ', 0, 1 \rangle$, sendo $\langle P, R \rangle$ a correspondente relação de ordem, pode-se afirmar que:

a) *Princípio da Consistência.* $a \times b = a$ se e somente se $a + b = b$ se e somente se $a + b$
b) se $a \times b = a \times c$ e $a + b = a + c$, então $b = c$
c) $(a \times b)\, R\, a$ e $a\, R\, (a + b)$
d) $0\, R\, a$ e $a\, R\, 1$

Exercício 10.29 Suponha uma *Álgebra Booleana Finita*, ou seja, cujo conjunto suporte é finito. Então, mostre que:

Dica: as provas não são triviais. Se o leitor tiver alguma dificuldade em desenvolver as provas, sugere-se fazer uma pesquisa;

a) O conjunto suporte possui 2^n elementos, para algum $n \in \mathbf{N}$;
b) É isomorfa a $\langle \mathbf{P}(A), \subseteq \rangle$, para algum conjunto A finito.

Exercício 10.30 Usando as portas E, OU e NÃO, desenhe os circuitos lógicos correspondentes a cada uma das seguintes expressões:

a) $(X' + Y) \bullet Z$
b) $(X + Y)' + X' \bullet Z$
c) $X' \bullet Y + (X \bullet Y)'$

Exercício 10.31 Na Figura 10.22 é esquematizado o somador *full-adder*, usando o somador mostrado no EXEMPLO - 10.30 - Circuito com Múltiplas Saídas: *half-adder*. O circuito *full-adder* adiciona 3 bits (X, Y e o *carry* C_{in}) e produz duas saídas (a soma S e o *carry* C_{out}). Então:

a) Desenhe detalhadamente o circuito;
b) Interprete o seu funcionamento.

Exercício 10.32 Os reticulados $\langle \mathbf{P}(\{a, b\}), \subseteq \rangle$ e $\langle \mathbf{P}(\{a, b\}), \supseteq \rangle$ são reticulados isomorfos?

Exercício 10.33 Apresente duas cadeias as quais são reticulados isomorfos e:

a) Prove que são reticulados isomorfos;
b) Apresente um homomorfismo que não seja isomorfismo.

Exercício 10.34 A relação $\langle D_{36}, : \rangle$ (introduzida no EXEMPLO 10.10 - Reticulado: Divisores) constitui um reticulado distributivo? Justifique a sua resposta.

Exercício 10.35 O reticulado ilustrado na Figura 10.23 (como diagrama de Hasse) é distributivo? Justifique a sua resposta.

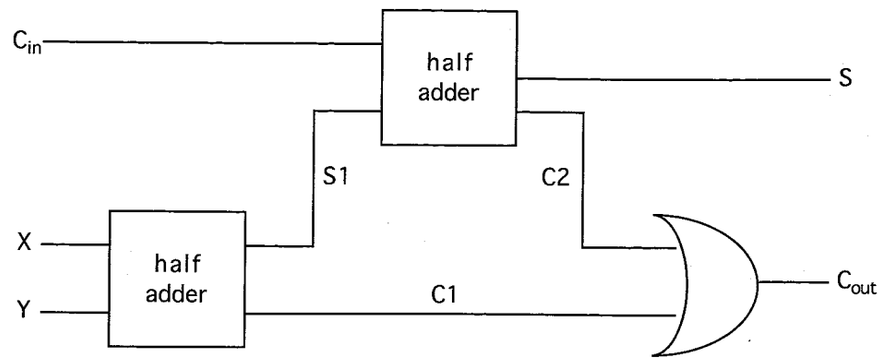


Figura 10.22 Circuito com múltiplas saídas: *full-adder*

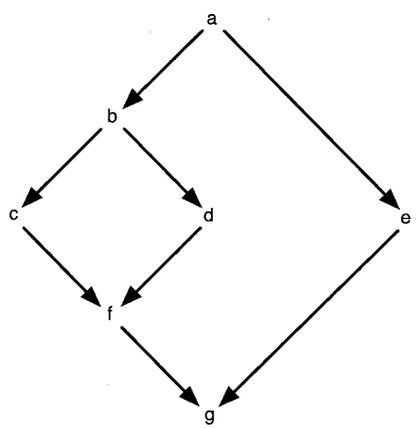


Figura 10.23 Diagrama de Hasse

Exercício 10.36 Em um homomorfismo Booleano, prove que a preservação das operações de complemento e de produto (respectivamente, de soma) implica a preservação da soma (respectivamente, do produto), do elemento inicial e do elemento terminal.

Exercício 10.37 Prove que a composição de homomorfismos Booleanos é um homomorfismo Booleano.

11 Conclusões

Este livro apresenta os principais conceitos de Matemática Discreta, de forma didática e acessível, de acordo com Diretrizes Curriculares do MEC para Cursos de Computação e Informática [MEC 2005]. Sempre que possível, as construções apresentadas são instanciadas em casos aplicados à Computação e Informática nas mais variadas matérias e disciplinas como, por exemplo, *Sistemas Operacionais*, *Bancos de Dados*, *Compiladores*, *Estruturas de Dados*, *Técnicas Digitais*, *Algoritmos*, *Complexidade de Algoritmos*, *Teoria da Computação*, *Linguagens Formais e Autômatos*, *Modelos para Concorrência*, *Semântica Formal*, *Teoria das Categorias*, *Programação*, *Paradigmas de Linguagens de Programação*, etc. Em experiências realizadas seguindo esta abordagem, ficou claro que, além de o leitor fixar muito mais facilmente os principais conceitos e resultados desenvolvidos, enfrenta com grande naturalidade diversos estudos subseqüentes, resultando em um aproveitamento muito mais efetivo do curso.

Embora o texto seja didático e acessível, não descuida do desenvolvimento do raciocínio nem dos aspectos matemático-formais. Pode ser usado como livro-texto, ou como livro de referência para os mais diversos aspectos da Computação e Informática. Adicionalmente, embora o conteúdo básico da matéria Matemática Discreta seja relativamente estável (comparativamente com e evolução tecnológica da Computação e Informática), a abordagem dá ênfase às questões e aos problemas da atualidade, bem como às novas abordagens dos *Fundamentos da Computação*, como as inspiradas em Teoria das Categorias.

Assim, o leitor que acompanhou satisfatoriamente o conteúdo tratado ao longo deste livro, além de:

a) Desenvolver a sua capacidade de raciocínio abstrato (lógico-matemático) como um todo;
b) Obter uma visão abrangente de uma parte significativa da Computação e Informática;

deve ser capaz de:

c) Aplicar os conceitos básicos da Matemática Discreta como uma ferramenta Matemática para investigações e aplicações precisas em Computação e Informática;
d) Via Matemática Discreta, abordar problemas aplicados e enfrentar ou propor com naturalidade novas tecnologias.

A seqüência natural e esperada de continuidade dos estudos relacionados com Matemática Discreta é a seguinte:

- *Análise Combinatória* e *Probabilidade Discreta* (temas não abordados);
- *Teoria dos Grafos* (apenas alguns tópicos são apresentados);
- *Teoria das Categorias*.

Em particular, Teoria das Categorias, em muitos aspectos, lembra e se assemelha aos estudos desenvolvidos em Matemática Discreta, com a vantagem de que permite abordar problemas possivelmente complexos de forma mais simples. De fato, a expressividade das construções categoriais tem sido uma das principais (senão a principal) motivações para o uso da Teoria das Categorias na Computação e Informática. Considerando-se a complexidade dos sistemas

computacionais atuais, verifica-se que, de certa forma, o desenvolvimento de soluções para os problemas propostos está limitado à capacidade do ser humano de expressar os problemas e suas soluções. Assim, quanto mais expressivo for o formalismo usado, mais avanços podem ser esperados. Inclusive, formalismos mais expressivos auxiliam não só nas especificações e provas, mas, principalmente, em um melhor entendimento dos problemas, bem como em uma maior simplicidade e clareza nas soluções. Por fim, registre-se que, Jean Piaget, em trabalho publicado após seu falecimento, afirma que:

Teoria das Categorias reflete a constituição genética
das ferramentas cognitivas do homem

12 Bibliografia

[Aho & Ullman 1972] Aho, A. V. e Ullman, J. D., *The Theory of Parsing, Translation and Compiling*, Prentice-Hall, 1972.

[Barr & Wells 1990] M. Barr & C. Wells, *Category Theory for Computing Science*, Prentice Hall.

[Cláudio *et al* 1987] D. M Cláudio, T. A. Diverio & L. V. Toscani, *Fundamentos da Matemática Computacional*, Sagra-Luzzatto.

[Diverio & Menezes 2000] T. A. Diverio & P. B. Menezes, *Teoria da Computação: Máquinas Universais e Computabilidade*, Sagra-Luzzatto.

[Diverio & Menezes 2001] T. A. Diverio & P. B. Menezes, *Teoria da Computação e os Profissionais da Área de Computação*, IX Escola Regional de Informática, SBC.

[Domingues & Iezzi 1982] H. H. Domingues & G. Iezzi, *Álgebra Moderna*, terceira edição, Atual.

[Dornhoff 1978] L Dornhoff, *Applied Modern Algebra*, Macmillan.

[Gerstin 2003] J. L. Gerstin, *Mathematical Structures for Computer Science*, quinta edição, W. H. Freeman.

[Graham *et al* 1994] R. L. Graham, D. E. Knuth, O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*, segunda edição, Addison-Wesley.

[Heuser 2001] C. A. Heuser. *Projeto de Banco de Dados*, quarta edição, Sagra-Luzzatto.

[Hennessy 1988] M. Hennessy, *Algebraic Theory of Processes*, MIT Press.

[Hoare 1985] C. A. R. Hoare, *Communicating Sequential Processes*, Prentice Hall.

[Hopcroft & Ullman 2000] J. E. Hopcroft & J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, segunda edição, Addison-Wesley.

[Korfhage 1997] R. R. Korfhage, *Discrete Computational Structures*, segunda edição, Academic Press.

[Lawvere & Rosebrugh 2002] W. Lawvere & R. Rosebrugh, *Sets for Mathematics*, Cambridge University Press.

[Laufer 1984] H. B. Laufer, *Discrete Mathematics and Applied Modern Algebra*, Prindle Weber & Schmidt.

[Levy 1980] L. S. Levy, *Discrete Structures of Computer Science*, John Wiley and Sons.

[Mac Lane 1998] S. Mac Lane, *Categories for the Working Mathematician*, segunda edição, Springer-Verlag.

[Manna 1974] Z. Manna, *Mathematical Theory of Computation*, McGraw-Hill, 1974.

[MEC 2005] *Diretrizes Curriculares para Cursos de Computação e Informática.* http://www.mec.gov.br/Sesu/diretriz.shtm

[Menezes 2005] P. B. Menezes, *Linguagens Formais e Autômatos*, quinta edição, Sagra-Luzzatto.

[Menezes & Haeusler 2001] P. B. Menezes & H. E. Haeusler, *Teoria das Categorias para Ciência da Computação*, Sagra-Luzzatto.

[Menezes *et al* 2000] P. B. Menezes, L. V. Toscani, T. A. Diverio, L. Ribeiro, L. C. Zeni, M. M. Sieczkowski, *Uma Proposta de Plano Pedagógico para a Matéria de Matemática*, II Curso de Qualidade de Cursos de Graduação da Área de Computação e Informática, p.65 – 102, Editora Universitária Champagnat.

[Menezes *et al* 2001] P. B. Menezes, R. F. Weber, H. E. Haeusler, A. C. V. Melo, M. S. Camargo, *Proposta de Plano Pedagógico para Cursos de Ciência da Computação*, III Curso de Qualidade de Cursos de Graduação da Área de Computação e Informática, v.1, p.335 – 392, SBC.

[Menezes *et al* 1998] P. B. Menezes, D. J. Nunes, T. A. Diverio, L. Ribeiro, V. Rodrigues, L. V. Toscani, *Desenvolvimento da Área Formal da Computação no Instituto de Informática da UFRGS*, I Workshop de Métodos Formais, p.1 – 12, UFRGS, 1998.

[Meseguer & Montanari 1990] J. Meseguer & U. Montanari, *Petri Nets are Monoids*, Information and Computation 88, pp. 105-155, Academic Press.

[Milner 1980] R. Milner, *A Calculus for Communicating Systems*, LNCS 92, Springer-Verlag.

[Milner 1989] R. Milner, *Communication and Concurrency*, Prentice Hall.

[Munro 1992] J. E. Munro, *Discrete Mathematics for Computing*, Chapman & Hall.

[Nielsen *et al* 1994] M. Nielsen, V. Sassone & G. Winskel, *Relationship Between Models for Concurrency*, LNCS 803, Springer-Verlag , p. 425-476.

[Olderog 1991] E. R. Olderog, *Nets Terms and Formulas*, Cambridge Tracts in Theoretical Computer Science 23, Cambridge University Press.

[Oliveira *et al* 2001] R. S. Oliveira, A. S. Carissimi & S. S. Toscani, *Sistemas Operacionais*, segunda edição, Sagra-Luzzatto.

[Piaget 1992] J. Piaget. *Morphisms and Categories: Comparing and transforming*. Laurence Erlbaum Associates, Inc., New Jersey, 1992.

[Prather 1976] R. E. Prather. *Discrete Mathematical Structures for Computer Science*. Houghton Mifflin.

[Preparata & Yeh 1973] F. P. Preparata & R. Yeh. *Introduction to Discrete Structures for Computer Science and Engineering*. Addison Wesley.

[Reis 2002] R. A. L. Reis, *Concepção de Circuitos Integrados*, segunda edição, Sagra-Luzzatto.

[Price & Toscani 2001] A. M. A. Price & S. S. Toscani, *Implementação de Linguagens de Programação: Compiladores*, segunda edição, Sagra-Luzzatto.

[Reisig 1985] W. Reisig, *Petri Nets: An Introduction*, EATCS Monographs on Theoretical Computer Science 4, Springer-Verlag.

[Rosen 2003] K. H. Rosen, *Discrete Mathematics and Its Applications*, quinta edição, McGraw-Hill.

[Rosen *et al* 1999] K. H. Rosen (ed.), J. G. Michaels (ed.), J. L. Gross (ed.), J. W. Grossman (ed.) & D. R. Shier (ed.), *Handbook of Discrete and Combinatorial Mathematics*, CRC Press.

[Sassone *et al* 1993] V. Sassone, M. Nielsen & G. Winskel, *A Classification of Models for Concurrency*, LNCS 715, Springer-Verlag, p. 82-96.

[Scott 1976] D. S. Scott, *Data Types as Lattices*, SIAM Journal of Comput. No. 5, Vol. 3.

[Scott 1982] D. S. Scott, *Domains for Denotational Semantics*, LNCS, Vol. 140, Springer-Verlag.

[Sernadas 1992] C. Sernadas, *Introdução à Teoria da Computação*, Editora Presença, Portugal.

[Stoy 1981] J. E. Stoy, *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*, MIT Press.

[Toscani & Veloso 2001] L. V. Toscani & P. A. S. Veloso, *Complexidade de Algoritmos: Análise, Projeto e Métodos*, Sagra-Luzzatto.

[Toscani *et al* 2003] S. S. Toscani, R. S. Oliveira & A. S. Carissimi, *Sistemas Operacionais e Programação Concorrente*, Sagra-Luzzatto.

[Tremblay & Monahar 1975] J. P. Tremblay & R. Monahar, *Discrete Mathematical Structures With Applications to Computer Science*, McGraw Hill.

[Walters 1991] R. F. C. Walters, *Categories and Computer Science*, Cambridge University Press.

[Weber 2004] R. F. Weber, *Fundamentos de Arquitetura de Computadores*, terceira edição, Sagra-Luzzatto.

[Winskel 1987] G. Winskel, *Petri Nets, Algebras, Morphisms and Compositionality*, Information and Computation 72, Academic Press, pp. 197-238.

[Winskel 1987b] G. Winskel, *Event Structures*, LNCS 255, Springer-Verlag, pp. 325-392.

[Winskel 1988] G. Winskel, *An Introduction to Event Structures*, LNCS 354, Springer-Verlag, pp. 29-95.

Índice Remissivo

1

1 (categoria) 207
1+1 (categoria) 207

2

2 (categoria) 207

A

Abeliana (álgebra) 188
Abeliano (grupo) 189
Abeliano (grupóide) 188, 189
Abeliano (monóide) 189
Abeliano (semigrupo) 189
Absorção (conjuntos) 38, 46, 64
Absorção (lógica) 35, 38
Absorção (reticulado) 221, 232
Absurdo (prova) 31
Ackernmann (função) 183
Alfabeto 5
Álgebra 37, 185, 188
Álgebra Booleana 56, 213, 232, 234
Álgebra Booleana Finita 244
Álgebra de Boole 56, 213, 232, 234
Álgebra de Conjuntos 37, 185, 233
Álgebra de Funções 123, 185, 210
Álgebra de Funções Parciais 185
Álgebra de Processos 99
Álgebra de Proposições 185
Álgebra de Relações 92, 185
Álgebra Grande 41, 185
Álgebra Interna 188
Álgebra Monossortida 202
Álgebra Pequena 41, 185
Álgebra Polissortida 202
Algoritmo 13, 30, 152, 156, 170, 173
Algoritmo (complexidade) 134
Análise Combinatória 1, 245
Análise Léxica 6
Análise Semântica 6
Análise Sintática 6
Anti-Reflexiva (relação) 125, 127
Aplicação 105
Arco (grafo) 71, 202
Arcos Paralelos (grafo) 92
Aresta (grafo) 71, 202
Aritmética Computacional 196
Arquitetura de Computadores 8, 196
Arquivo de Acesso Direto 114
Arranjo (tipo de dado) 134
Associativa (álgebra) 186, 187
Associativa (categoria) 206
Associativa (intersecção) 38, 45, 64
Associativa (lógica) 34, 38
Associativa (reticulado) 232
Associativa (união) 38, 43
Atômica (proposição) 14
Átomo 14
Autômato Finito 95, 100, 153
Auto-Relação 69
Avaliação Procrastinada (programação funcional) 119

B

Back Tracking 47
Backus Naur Form 172
Banco de Dados 84
Banco de Dados Relacional 67, 84
Base (sistema numérico) 196
Base Binária 196
Base de Indução 160
Base Decimal 196
Base Unária 196
Bicondição 18
Bijetora (função) 93, 105
Bit 196
Bloco da Partição 139
Bloco-Estruturada (linguagem) 156
BNF 160, 172
Boa-Ordem (relação) 159
Bool (categoria) 240
Boole (álgebra) 56, 213, 232, 234
Boole (lógica) 14
Boole, George 14, 213
Booleana (álgebra) 56, 213, 232, 234
Booleano (tipo de dado) 8
Bubblesort 134

C

C (linguagem de programação) 6, 152
C.P.O. 213
Cabeça da Fita (autômato) 100
Cabeça da Fita (máquina de Turing) 154
Cadeia 132
Cadeia de Caracteres 5
Cadeia Estrita 132
Cadeia Vazia 5
Cálculo Diferencial e Integral 2
Caminho (grafo) 129
Cancelamento (grupo) 192
Cancelamento à Direita (grupo) 192
Cancelamento à Esquerda (grupo) 192
Cantor (diagonalização) 149
Cantor (teorema) 151
Cantor, G. 149, 152
Caractere 5

Caractere (tipo de dado) 8
Cardinal 151
Cardinalidade (conjunto) 83, 147, 148
Cardinalidade Finita 148
Cardinalidade Infinita 148
Categoria 185, 204, 205
Categoria 1 207
Categoria 1+1 207
Categoria 2 207
Categoria Bool 240
Categoria Gr 206
Categoria Lattice 239
Categoria Mon 206
Categoria Poset 238
Categoria Set 206
Categoria Vazia 207
Church (hipótese) 13, 30, 170
Church, A. 30
Ciência da Computação 204
Circuito Lógico 213, 234
Classe de Equivalência 139
Classificação de Dados 125, 133
Codomínio (relação) 68
Cohen 152
Colisão (função de hashing) 115
Compilador 6, 100, 137
Compiladores (disciplina) 5
Complemento (conjunto) 37, 48
Complemento (reticulado) 226, 233, 234
Complexidade de Algoritmos 104, 134
Composição (categoria) 204, 206
Composição (relações) 67, 75
Composição Paralela 138
Composta (relação) 73
Compreensão (denotação por) 3
Computabilidade 95
Computação Atômica (autômato) 101, 154
Computação Atômica (rede de Petri) 87
Computações (autômato finito) 160, 169
Comutativa (álgebra) 186, 187
Comutativa (intersecção) 38, 45, 64
Comutativa (lógica) 34, 38
Comutativa (reticulado) 232, 234
Comutativa (união) 38, 43, 64
Concatenação 95, 106
Conclusão (condição) 17
Concorrência 87, 125, 137, 213, 229
Concorrência Verdadeira 137
Condição 18
Conetivo (lógico) 14
Conetivo Contradição 210
Conetivo E 16
Conetivo EXOR 35
Conetivo NAND 35
Conetivo Não 15
Conetivo NOR 210
Conetivo Ou 17
Conetivo Se-Então 18
Conetivo Se-Somente-Se 18
Conetivo Tautologia 210
Conexa (relação) 131
Conjunção 16
Conjunto 2
Conjunto Contável 2, 149
Conjunto das Partes 37, 51
Conjunto de Chegada (relação) 68
Conjunto de Índices 114
Conjunto de Partida (relação) 68
Conjunto dos Números Inteiros 4
Conjunto dos Números Irracionais 4
Conjunto dos Números Naturais 4
Conjunto dos Números Racionais 4
Conjunto dos Números Reais 4
Conjunto Enumerável 148
Conjunto Falsidade 26
Conjunto Finitamente Contável 148
Conjunto Finito 1, 5, 147, 148
Conjunto Gerador (monóide) 201
Conjunto Imagem (relação) 70
Conjunto Indexado 95, 114
Conjunto Infinitamente Contável 148
Conjunto Infinito 5, 147, 148
Conjunto Não-Contável 2, 148
Conjunto Ordenado 132, 213
Conjunto Ordinário 40
Conjunto Parcialmente Ordenado 213
Conjunto Potência 51
Conjunto Quociente 142
Conjunto Suporte (álgebra) 188
Conjunto Unitário 4
Conjunto Universo 7
Conjunto Vazio 4
Conjunto Verdade 26
Conjuntos Disjuntos 45
Conjuntos Equipotentes 150
Conjuntos Independentes 45
Conjuntos Isomorfos 82
Conjuntos Mutuamente Exclusivos 45
Constante (função) 106
Contável (conjunto) 2, 149
Contém 7
Contém Propriamente 7
Contido 6
Contido Amplamente 7
Contido Estritamente 7
Contido Propriamente 7
Continência 6
Continuum 152
Continuum (hipótese) 152
Contradição 23, 26
Contradição (conetivo) 210
Contradomínio (relação) 68
Contraposição 24
Contraposição (prova) 31
Coordenadas polares 70
Coproduto 214
Corolário 29

D

Definição Indutiva 19, 130, 160, 165
Definição Recursiva 130, 160, 165
Definida (relação) 69
Demonstração (teorema) 13
Demonstração Direta 31
Demonstração por Absurdo 31, 32
Demonstração por Contra-Exemplo 32
Demonstração por Contraposição 31, 32
Demonstração por Indução 31
Demonstração por Redução ao Absurdo 31, 32
DeMorgan (conjuntos) 38, 49, 64
DeMorgan (lógica) 34, 38
DeMorgan (reticulado) 233, 244
DeMorgan, Augustus 34
Denotação por Compreensão 3
Denotação por Extensão 3
Dependência Causal 229
Derivação (gramática) 170, 171
Destino (categoria) 205
Destino (grafo) 202
Destino (relação) 68
Diagonalização de Cantor 149
Diagrama de Hasse 136
Diagrama de Venn 37, 39, 69
Diagrama Entidade-Relacionamento 85
Diagrama E-R 85
Diferença (conjuntos) 37, 49
Direta (prova) 31
Disjunção 17
Distributiva (intersecção sobre a união) 38, 46
Distributiva (lógica) 34, 38
Distributiva (produto cartesiano sobre a intersecção) 53, 66
Distributiva (produto cartesiano sobre a união) 53, 66
Distributiva (reticulado) 233, 234
Distributiva (união sobre a intersecção) 38, 46, 64
Domínio (relação) 68
Domínio de Definição (relação) 70
Domínio de Valores (relação) 70
Dual (grafo) 74
Dual (relação) 67, 73
Dupla Negação 34, 38
Duplo Complemento (conjuntos) 38, 48
Duplo Complemento (reticulado) 233, 244

E

E (conetivo) 16
E (porta) 235
Elemento 2
Elemento Absorvente (conjuntos) 64
Elemento Absorvente (reticulado) 226, 233
Elemento Inicial 214, 215
Elemento Inicial (reticulado) 234
Elemento Inverso (álgebra) 186, 187
Elemento Neutro (álgebra) 186, 187
Elemento Neutro (intersecção) 45, 64
Elemento Neutro (reticulado) 226, 233, 234
Elemento Neutro (união) 43, 64
Elemento Neutro à Direita (álgebra) 187
Elemento Neutro à Esquerda (álgebra) 187
Elemento Terminal 214, 215
Elemento Terminal (reticulado) 234
Elemento Zero 216
Endoarco 135, 145
Endoaresta 135, 145
Endorrelação 69
Engenharia de Software 13, 103
Entidade (diagrama E-R) 86
Enumeração 2, 148
Enumerável (conjunto) 148
Epimorfismo 67, 77, 81
Epimorfismo (estruturas algébricas) 193, 236
Epirrelação 81
Equipotentes (conjuntos) 150
Equivalência (relação) 23, 24
Escolha 88, 138
Estado (autômato) 101
Estado (máquina de Turing) 154
Estado Final (autômato) 101
Estado Final (máquina de Turing) 154
Estado Inicial (autômato) 101
Estado Inicial (máquina de Turing) 154
Estrutura de Eventos 138
Estruturas de Dados 115
Existencial (quantificador) 27
EXOR (conetivo) 35, 116
Expressão Regular 160, 167
Extensão (denotação por) 3

F

Falso (valor-verdade) 14
Fechada (operação) 51
Fecho (relação) 125, 129
Fecho de Kleene 185, 201
Fecho Reflexivo 130
Fecho Reflexivo e Transitivo 130
Fecho Simétrico 130
Fecho Transitivo 130, 165
Finitamente Contável (conjunto) 148
Finito (conjunto) 1, 5, 147, 148
Fita (autômato) 100
Fita (máquina de Turing) 154
Fonte (linguagem) 6
Fonte (rede de Petri) 90
Fork (primitiva) 229
Forma de Backus Naur 160, 172
Fórmula 19, 165
Fórmula (lógica) 19, 165
Função 59, 67, 78, 93, 95, 105
Função Bijetora 93, 105
Função Computável 30, 170, 173
Função Concatenação 106
Função Constante 106
Função de Ackernmann 183
Função de Aleatorização 115
Função de Cálculo de Endereço 115

Função de Hashing 115
Função de Randomização..... 115
Função de Transição (autômato)..... 100
Função de Transição (máquina de Turing)... 154
Função Imersão.....107, 108
Função Inclusão.....106, 107
Função Monotônica.....204, 236, 237
Função Parcial..... 67, 95, 96, 105
Função Programa (autômato) 100, 102
Função Programa (máquina de Turing).154, 156
Função Programa Estendida (autômato finito) 169
Função Projeção.....107, 108
Função Recursiva de Kleene160, 173, 175, 179
Função Recursiva Parcial 160, 175, 179
Função Recursiva Primitiva..... 183
Função Total..... 95
Funcional (linguagem de programação)..... 117
Funcional (programação)..... 117
Funcional (relação) 67, 77, 78
Fundamentos da Computação 245

G

Geração de Código..... 6
Gerador (monóide) 201
Gödel 152
Gr (categoria)..... 206
Grafo 71, 185, 202
Grafo Dual..... 74
Grafo Pequeno Direcionado..... 202
Gramática..... 170
Gramática de Chomsky..... .....137, 160, 169, 170, 173
Grande (álgebra).....41, 185
Grande (estrutura matemática)..... 41
Grupo.....188, 189
Grupo Abeliano..... 189
Grupo Comutativo 189
Grupóide 188
Grupóide Abeliano 188, 189
Grupóide Comutativo..... 189

H

Haskell (linguagem de programação)..... 117
Hasse (diagrama)..... 136
Hasse, Helmut..... 136
Hipótese..... 29
Hipótese de Church 13, 30, 170, 173, 175
Hipótese de Indução 160
Hipótese do Continuum..... 152
Homomorfismo..... 185
Homomorfismo Booleano 240
Homomorfismo de Álgebras..... 185
Homomorfismo de Álgebras Booleanas 236, 240
Homomorfismo de Conjuntos Parcialmente Ordenados 236, 237
Homomorfismo de Grafos..... 203
Homomorfismo de Grupóides..... 193
Homomorfismo de Grupos 199
Homomorfismo de Monóides.....197
Homomorfismo de Reticulados 236, 238
Homomorfismo de Semigrupos.....193, 197

I

Idempotência (intersecção)..... 38, 45, 64
Idempotência (lógica) 34, 38
Idempotência (reticulado) 223, 232
Idempotência (união)..... 38, 43, 64
Identidade (categoria).....206
Identidade (relação).....82
Iguais a menos de Isomorfismo.....84
Iguais a menos de Isomorfismo (estruturas algébricas).....193, 237
Igualdade (conjuntos).....6, 7
Imagem (conjunto) 70
Imagem (relação).....69
Imersão (função) 107, 108
Implicação (relação) 23
Inclusão (função).....106, 107
Independência (concorrência) 138, 229, 230
Índices (conjunto).....114
Indução (base).....160
Indução (hipótese).....160
Indução (passo) 160
Indução (prova)..... 31, 160, 161
Indução em Estrutura 164
Indução Estruturada.....164
Indução Estrutural.....164
Indução Matemática (primeiro princípio)160, 163
Indução Matemática (princípio).....159, 160
Indução Matemática (segundo princípio).....163
Indução Matemática Finita (princípio) 183
Indutiva (definição).....160, 165
Indutiva (prova) 160, 161
Indutivamente Definido 165
Ínfimo.....214
Infinitamente Contável (conjunto).....148
Infinito (conjunto)..... 5, 147, 148
Inicial (elemento).....215
Inicial (reticulado).....234
Injetora (relação)..... 67, 77, 78
Inteiro (tipo de dado) 8
Inteiros (conjuntos) 4
Intersecção.....37, 44
Intersecção de Funções Parciais.....124
Intersecção de Funções Totais 124
Intervalo 243
Intervalo Aberto 243
Intervalo Fechado 243
Inversa (relação).....73, 82
Inversa à Direita (relação).....82
Inversa à Esquerda (relação) 82
Inversor.....235
Irracionais (conjunto) 4
Irreflexiva (relação).....125
Isomorfas (estruturas algébricas) 193, 236
Isomorfismo..... 67, 77, 82
Isomorfismo (estruturas algébricas)..... 193, 236

Isomorfismo (grupóides) 195
Isomorfismo (grupos)..... 199
Isomorfismo (monóides)..... 198
Isomorfismo (semigrupos)..... 197
Isomorfos (reticulados)..... 239
Isorrelação 82

J

Java (linguagem de programação)..... 6
Join (primitiva) 230

K

Kleene (fecho).....185, 201
Kleene (função recursiva) ... 160, 173, 175, 179
Kleene, S. C. 175

L

Lattice (categoria)..... 239
Lazy Evaluation (programação funcional)..... 119
Lema..... 29
Lexicográfica (ordem) 133
Limitante Inferior..... 214
Limitante Superior..... 214
Linguagem 5, 6, 8
Linguagem Bloco-Estruturada..... 156
Linguagem de Programação Funcional..95, 117
Linguagem de Programação Funcional Pura 117
Linguagem Fonte..... 6
Linguagem Formal.....6, 8, 201
Linguagem Funcional..... 174
Linguagem Gerada (gramática) 171
Linguagem Imperativa.....119, 174
Linguagem Lógica.....19, 165
Linguagem Não-Computável..... 60
Linguagem Natural 170
Linguagem Objeto 6
Linguagem Recursiva..... 60
Linguagem Recursivamente Enumerável..... 60
Linguagem Regular.....104, 167
Linguagens Formais..5, 37, 100, 137, 170, 185
Livre (monóide)..... 201
Livremente Gerado (monóide)..... 201
Lógica (matemática)..... 13, 14, 233
Lógica Booleana 14
Lógica de Boole..... 14
Lógico (tipo de dado)..... 8
Lugar (rede de Petri)..... 87

M

Maior Elemento.....214, 215
Maior Limitante Inferior..... 214
Mapeamento Parcial 96
Máquina de Turing 30, 60, 147, 152, 153, 156, 170, 173
Marcação (rede de Petri) 87
Matemática Computacional..... 8
Matemática Discreta..... 1, 2, 147, 153
Matemática do Continuum..... 2
Matriz Transposta 74
Menor Elemento.....214, 215
Menor Limitante Superior 214
Minimização (função recursiva parcial)..177, 179
Modelo Conceitual (banco de dados) 85
Mon (categoria).....206
Mônada (linguagem de programação funcional) 118
Mônada (Teoria das Categorias) 118
Monóide.....188, 189
Monóide Abeliano.....189
Monóide Comutativo.....189
Monóide Livre.....201
Monóide Livre Gerado.....201
Monóide Livremente Gerado 201
Monomorfismo..... 67, 77, 80
Monomorfismo (estruturas algébricas) .193, 236
Monorrelação.....80, 81
Monossortida (álgebra).....202
Morfismo 185
Morfismo (categoria).....205
MOS (Metal Óxido Semicondutor).....236
Multiconjunto..... 95, 111, 112
Multigrafo 203

N

NAND (conetivo).....35, 123
NAND (porta) 236
Não (conetivo) 15
Não (porta).....235
Não-Computável (linguagem).....60
Não-Contável (conjunto).....2, 148
Não-Determinismo 88, 138
Não-Solucionável (problema).....152
Não-Terminal (gramática).....170
Naturais (conjunto).....4
Negação.....15
Nodo (grafo) 71, 202
NOR (conetivo).....210
Núcleo (sistema operacional) 229
Números Inteiros (conjunto).....4
Números Irracionais (conjunto).....4
Números Naturais (conjunto).....4
Números Racionais (conjunto) 4
Números Reais (conjunto) 4

O

Objeto (categoria) 204, 205
Objeto (linguagem) 6
Operação 186
Operação Binária.....186
Operação Binária Interna.....186
Operação de Composição (categoria)..204, 206
Operação Destino (categoria).....205
Operação Destino (grafo).....202
Operação Fechada.....51, 186
Operação Identidade (categoria) 206
Operação Interna 186
Operação Não-Reversível (conjuntos).....37
Operação Origem (categoria) 205
Operação Origem (grafo) 202

Operação Parcial ... 96
Operação Reversível (conjuntos) ... 37
Operador Lógico ... 14
Oposta (relação) ... 73
Ordem (relação) ... 131
Ordem Conexa (relação) ... 132
Ordem Conexa Ampla (relação) ... 132
Ordem Conexa Estrita (relação) ... 132
Ordem Lexicográfica ... 133
Ordem Parcial (relação) ... 132, 213
Ordem Parcial Ampla (relação) ... 132, 213
Ordem Parcial Estrita (relação) ... 132
Ordenado (conjunto) ... 132, 213
Ordinário (conjunto) ... 40
Orientação a Objetos ... 103
Origem (categoria) ... 205
Origem (grafo) ... 202
Origem (relação) ... 68
Otimização de Código ... 6
Ou (conetivo) ... 17
Ou (porta) ... 235
Ou-Exclusivo (conetivo) ... 116

P

Palavra ... 5
Palavra Vazia ... 5
Palíndromo ... 6
Par Ordenado ... 52
Paradoxo de Russell ... 37, 40, 151
Parâmetro Atual (Pascal) ... 116
Parâmetro Formal (Pascal) ... 116
Parcialmente Ordenado (conjunto) ... 213
Parnas, D. ... 13
Partição (conjunto) ... 139
Pascal (linguagem de programação) ... 6, 8, 137, 152, 156, 201, 229
Passo de Indução ... 160
Pequena (álgebra) ... 41, 185
Pequena (estrutura matemática) ... 41
Pertence ... 4
Piaget, Jean ... 246
Plano Cartesiano ... 70
Polissortida (álgebra) ... 202
Ponto Flutuante (tipo de dado) ... 8
Porta E ... 235
Porta NAND ... 236
Porta Não ... 235
Porta Ou ... 235
Poset ... 132, 213
Poset (categoria) ... 238
Possui Inversa (relação) ... 73, 82
Possui Inversa à Direita (relação) ... 82
Possui Inversa à Esquerda (relação) ... 82
Premissa (condição) ... 17
Primeiro Princípio da Indução Matemática ... 160, 163
Primitiva ... 229
Primitiva (programação concorrente) ... 213
Primitiva (recursão) ... 176, 179
Primitiva Fork ... 229
Primitiva Join ... 230
Primitiva Quit ... 229
Princípio da Consistência (álgebra Booleana) 244
Princípio da Indução Matemática ... 31, 159, 160
Princípio da Indução Matemática Finita ... 183
Probabilidade Discreta ... 1, 245
Problema da Parada ... 61
Problema Não-Solucionável ... 152
Problema Solucionável ... 147, 152
Procedimento Efetivo ... 30
Produção (gramática) ... 170
Produto ... 214
Produto (reticulado) ... 234
Produto Cartesiano ... 37, 52
Produto de Funções ... 124
Produto de Funções Parciais ... 124
Programa (autômato) ... 100
Programa (máquina de Turing) ... 154
Programação Concorrente ... 213
Programação Funcional ... 117
Projeção (função) ... 107, 108
Prolog (linguagem de programação) ... 13
Proposição ... 14
Proposição Atômica ... 14
Proposição Sobre um Conjunto ... 25
Prova Direta ... 31
Prova Indutiva ... 160, 161
Prova por Absurdo ... 31, 32
Prova por Contra-Exemplo ... 32
Prova por Contraposição ... 31, 32
Prova por Indução ... 31, 51, 160, 161
Prova por Redução ao Absurdo ... 31, 32
Pura (linguagem de programação funcional) 117

Q

Quantificador Existencial ... 27
Quantificador Universal ... 27
Quit (primitiva) ... 229

R

Racionais (conjunto) ... 4
Reais (conjunto) ... 4
Real (tipo de dado) ... 8
Reconhecimento de Linguagem ... 60
Recursão ... 173
Recursão (linguagem de programação) 160, 173
Recursão Primitiva ... 176, 179
Recursiva (definição) ... 160, 165
Recursiva (linguagem) ... 60
Recursiva Parcial (função) ... 160, 175, 179
Recursiva Primitiva (função) ... 183
Recursivamente Definido ... 165
Recursivamente Enumerável (linguagem) ... 60
Rede de Petri ... 67, 87
Rede Lógica ... 213, 234
Redução ao Absurdo ... 25
Redução ao Absurdo (prova) ... 31
Reflexiva (relação) ... 125
Regra de Produção (gramática) ... 170

Relação ... 67, 68
Relação (como função) ... 111
Relação Anti-Reflexiva ... 125
Relação Anti-Simétrica ... 127
Relação Composta ... 73
Relação Conexa ... 131
Relação de Boa-Ordem ... 159
Relação de Derivação (gramática) ... 171
Relação de Equivalência ... 23, 24, 125, 139
Relação de Implicação ... 23
Relação de Ordem ... 125, 131
Relação de Ordem Conexa ... 132
Relação de Ordem Conexa Ampla ... 132
Relação de Ordem Conexa Estrita ... 132
Relação de Ordem Parcial ... 132, 213
Relação de Ordem Parcial Ampla ... 132, 213
Relação de Ordem Parcial Estrita ... 132
Relação Dual ... 67, 73
Relação em um Conjunto ... 69
Relação Funcional ... 78
Relação Identidade ... 82
Relação Injetora ... 78
Relação Inversa ... 73
Relação Irreflexiva ... 125
Relação Oposta ... 73
Relação Reflexiva ... 125
Relação Simétrica ... 125, 127
Relação Sobrejetora ... 79
Relação Total ... 79
Relação Transitiva ... 125, 128
Relacional (banco de dados) ... 84
Relacionamento (diagrama E-R) ... 86
Restrição do Contradomínio (função parcial) 120
Restrição do Domínio (função parcial) ... 99
Reticulado ... 213, 219, 223
Reticulado Complementado ... 224, 226
Reticulado Distributivo ... 224
Reticulado Limitado ... 224, 225
Reunião ... 42
Reuso de Software ... 103
Russell (paradoxo) ... 37, 40
Russell, Bertrand ... 40

S

Scheme (linguagem de programação) ... 118
Schröder-Bernstein (teorema) ... 150
Se-Então (conetivo) ... 18
Segundo Princípio da Indução Matemática ... 163
Semântica ... 137, 168, 170
Semântica Formal ... 100, 137
Semigrupo ... 188
Semigrupo Abeliano ... 189
Semigrupo Comutativo ... 189
Sentença ... 5
Sentença Vazia ... 5
Seqüência ... 95
Seqüência Finita ... 52, 113
Seqüência Infinita ... 113
Se-Somente-Se (conetivo) ... 18
Set (categoria) ... 206
Seta (categoria) ... 204, 205
Seta (grafo) ... 71, 202
Shannon, Claude ... 234
Símbolo ... 5
Símbolo Inicial (gramática) ... 170
Símbolo Não-Terminal (gramática) ... 170
Símbolo Terminal (gramática) ... 170
Símbolo Variável (gramática) ... 170
Simétrica (relação) ... 125, 127
Sincronização ... 138
Sintaxe ... 137, 168, 170
Sistema Biológico ... 170
Sistema Operacional ... 229
Sistemas Concorrentes e Comunicantes ... 138
SML (linguagem de programação) ... 118
Sobrejetora (relação) ... 67, 77, 79
Solucionável (problema) ... 147, 152
Soma ... 214
Soma (reticulado) ... 234
Sort ... 133
Sort (bubblesort) ... 134
Sorte (álgebra) ... 202
Subconjunto ... 6, 7
Subconjunto Próprio ... 7
Sub-Reticulado ... 227
Substituição (funções recursivas parciais) ... 175, 179
Sumidouro (rede de Petri) ... 90
Supremo ... 214

T

Tabela ... 114
Tabela-Verdade ... 15, 20
Tábua (operação) ... 208
Tautologia ... 23, 26
Tautologia (conetivo) ... 210
Técnicas de Demonstração ... 13, 24, 31
Técnicas Digitais ... 35, 55
Teorema ... 13, 23, 29
Teoria da Computação ... 30, 38, 59, 60, 170
Teoria da Concorrência ... 100
Teoria das Categorias ... 77, 80, 98, 118, 193, 204, 236, 245
Teoria dos Autômatos ... 37, 185
Teoria dos Conjuntos ... 2, 152
Teoria dos Grafos ... 1, 245
Terminal (elemento) ... 215
Terminal (gramática) ... 170
Terminal (reticulado) ... 234
Tese ... 29
Tipo de Dados ... 8
Token (rede de Petri) ... 87
Total (relação) ... 67, 77, 79
Transformação Parcial ... 96
Transição (autômato) ... 101
Transição (máquina de Turing) ... 154
Transição (rede de Petri) ... 87
Transição Habilitada (rede de Petri) ... 87

Transitiva (relação) 125, 128
Transposta (matriz) 74
Turing, A 30, 153

U

União 37, 42
União de Funções Parciais 124
União de Funções Totais 124
União Disjunta 37, 54, 138
Unidade de Controle (autômato) 100
Unidade de Controle (máquina de Turing) ... 154
Unitário (conjunto) 4
Universal (quantificador) 27
Universo (conjunto) 7
Upla Ordenada 52, 113

V

Valor-Verdade 14
Variável (gramática) 170
Variável Inicial (gramática) 170
Vazia (categoria) 207
Vazio (conjunto) 4
Venn (diagrama) 37, 39, 69
Venn, John 39
Verdadeiro (valor-verdade) 14
Vértice (grafo) 202

X

XML 119

Z

Zero (elemento) 216

Livraria: I.R. Ltda.
Compra/Verba: EACH/ 2006
Data: 20/02/2006
Preço: R$ 45,00